GPU Accelerated Intermixing as a Framework for Interactively Visualizing Spectral CT Data

---

A thesis submitted in partial fulfilment of the

requirements for the Degree of

Master of Engineering in Bioengineering

at the

University of Canterbury

by

Niels de Ruiter

University of Canterbury

2011

Abstract

Computed Tomography (CT) is a medical imaging modality which acquires anatomical data via the unique x-ray attenuation of materials. Yet, some clinically important materials remain difficult to distinguish with current CT technology. Spectral CT is an emerging technology which acquires multiple CT datasets for specific x-ray spectra. These spectra provide a fingerprint that allow materials to be distinguished that would otherwise look the same on conventional CT.

The unique characteristics of spectral CT data motivates research into novel visualization techniques. In this thesis, we aim to provide the foundation for visualizing spectral CT data. Our initial investigation of similar multi-variate data types identified intermixing as a promising visualization technique.

This promoted the development of a generic, modular and extensible intermixing framework. Therefore, the contribution of our work is a framework supporting the construction, analysis and storage of algorithms for visualizing spectral CT studies.

To allow evaluation, we implemented the intermixing framework in an application called MARSCTExplorer along with a standard set of volume visualization tools. These tools provide user-interaction as well as supportung traditional visualization techniques for comparison.

We evaluated our work with four spectral CT studies containing materials indistinguishable by conventional CT. Our results confirm that spectral CT can distinguish these materials, and reveal how these materials might be visualized with our intermixing framework.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Dr. Raphael Grasset, The HIT Lab NZ

Prof. Philip Butler, Physics and Astronomy Department, University of Canterbury.

Dr. Anthony P. H. Butler, The HIT Lab NZ and Director Bioengineering, University of Otago Christchurch.

Dr. Mike Hurrell, Radiology, University of Otago Christchurch.

Dr. Nigel Anderson, Radiology, University of Otago Christchurch.

The Visualization, Reconstruction, Gold Nanoparticle, Cartilage, Mice, and Human Excised Atheromatous Plaque Teams from the MARS Project.

University of Canterbury.

Family and friends.

Chapter I

Introduction

Medical imaging comprises techniques and processes used to detect disease, and to depict its relationship to the anatomy of that region. The aim is to provide sufficiently detailed data such that diagnosis is perfect regardless of the clinician.

Two factors contribute to the diagnostic value of a medical image; acquisition technology and visualization. Acquisition technology translates the real world into discrete data while visualization translates the data into a meaningful image for diagnosis by a clinician.

This research was conducted as part of the FRST Medipix All Resolution System (*MARS*) project (http://marsbioimaging.com/). The MARS project aims to develop and commercialize a new medical imaging modality called spectral computed tomography (*spectral CT*). The goal of this research was to provide the foundation for effective visualization of spectral CT data.

1.1 Spectral CT Imaging

Spectral CT is an x-ray imaging modality using the basic radiographical system shown in Fig. 1.1. A source sends x-rays through a subject to be captured by a detector on the other side. A projection image, such as Fig. 1.2, show the attenuation of x-rays by the subject's materials.

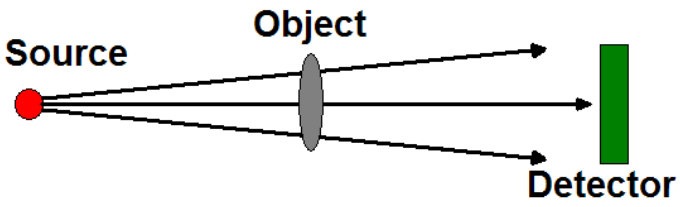

Figure 1.1: The basic system for all x-ray imaging modalities is based on radiography. A source sends x-rays through the subject to be detected on the other side.

Computed tomography (CT) extends radiography by acquiring a multitude of projection images at different orientations around the subject. A process called reconstruction then recreates the subject as a volumetric scalar dataset.

Over the years CT has advanced enough to identify and discriminate many important biological materials (e.g. calcium in bones, contrast agents in blood, etc.). Unfortunately, there are still other clinically important materials that CT can not distinguish due to similar attenuation behaviour such as iodine and barium. Recent technological developments have made spectral imaging feasible and have allowed the development of spectral CT imaging to solve this issue.

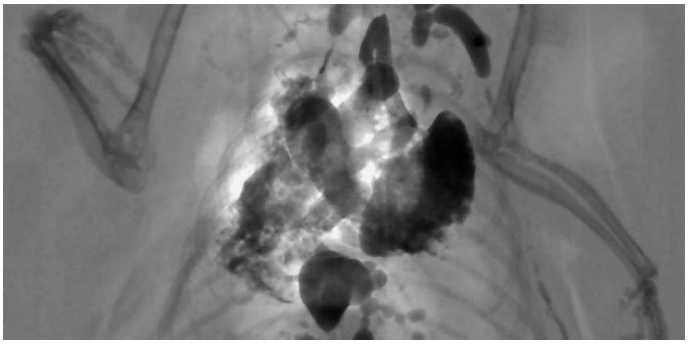

Figure 1.2: A projection image of a mouse thorax obtained with the new spectral CT scanner.

Figure 1.3: The current prototype of the MARS scanner, a novel spectral CT scanner.

The first implementation of spectral CT was called *Dual Energy CT* [2] which creates two CT datasets acquired with overlapping x-ray spectra. The differences between the two datasets (*energy bins*) provide additional material information. Unfortunately, Dual Energy CT is limited in its ability to identify all clinically important materials [2].

The MARS project led by the Physics and Astronomy Department of the University of Canterbury is developing a spectral CT scanner based on the Medipix family of detectors [3]. The current Medipix series, developed by CERN, is capable of acquiring a set of selected energy bins in parallel. The newest prototype, based on Medipix 3 and shown in Fig. 1.3, will be able to acquire up to eight energy bins simulataneously.

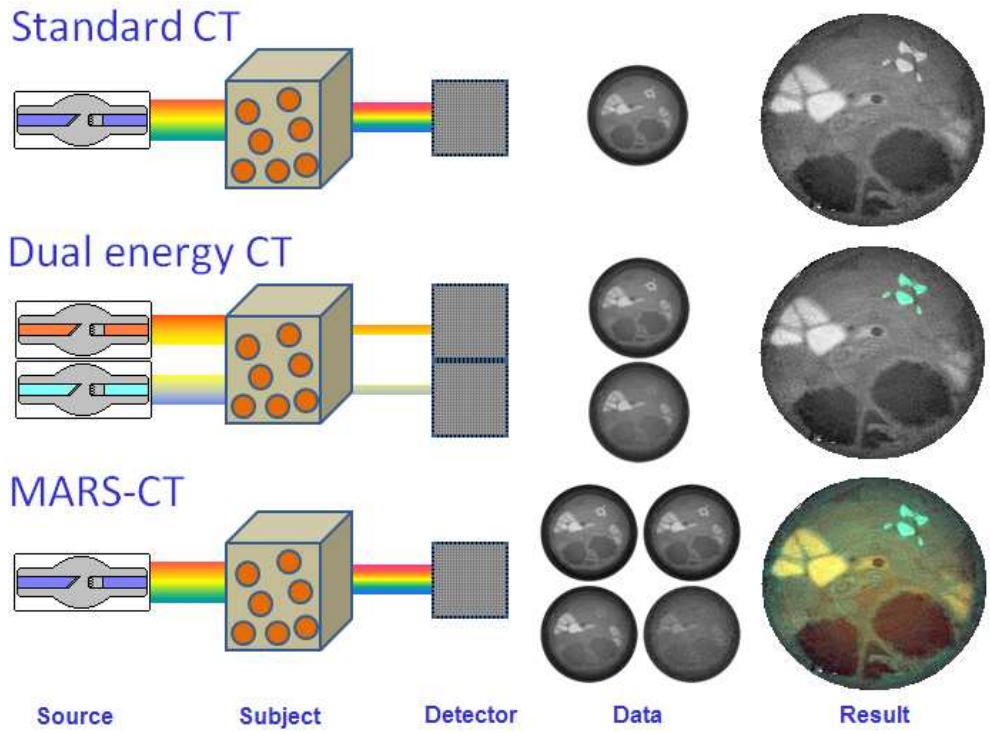

Figure 1.4: A simple illustration showing the difference between the three CT technologies.

The benefit of the MARS scanner is that acquiring selected energy bins allows specific materials to be targeted when their attenuation behaviour is known. This significantly increases the number of materials which can be identified or distinguised. Fig. 1.4 illustrates the key differences between the three CT technologies.

1.2 Spectral CT Visualization

The novelty of spectral CT imaging means that the visualization of its data is a new and open research field. The starting point of this research was an investigation into effective visualization techniques for similar data types. Spectral CT data is a multi-variate data type with similarities to multi-

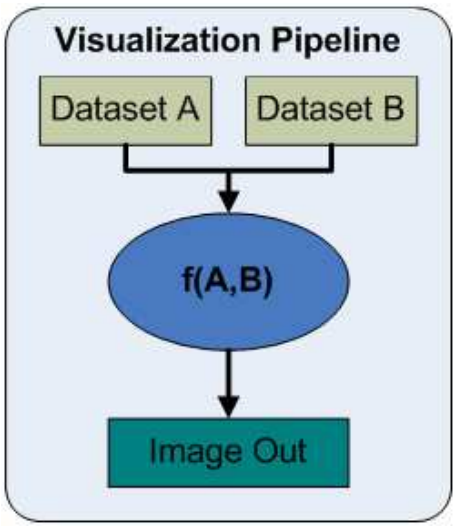

Figure 1.5: Intermixing combines multiple datasets at some stage in the visualization pipeline. The final result is a single visual outcome.

modal, multi-channel, and time-varying data.

A common approach for visualizating multi-variate data types is intermixing. Intermixing is the mathematical combination of data to produce a single visual outcome [4, 1]. The general process for intermixing is illustrated in Fig. 1.5.

Suitable algorithms for combining data are both application and task dependent. As the MARS scanner is still in development and potential applications (medical, biological) continue to emerge, suitable intermixing algorithms need to be developed for visualizing spectral CT data.

Algorithms for visualizing spectral CT data can be found through real-time algorithm construction and evaluation. This meant that the intermixing framework needed to be both generic and interactive. In addition, the intermixing framework also needed a visualization application to provide a comprehensive interface, to allow comparisons with traditional visualization

techniques, and to provide tools to examine and navigate through all results produced.

1.3 Research Objectives

The objectives of this research can then be summarized as follows.

- Design and implement a generic and flexible intermixing framework to construct and evaluate algorithms for combining energy bins in real-time.

- Design a structured visualization application around the intermixing framework to enable the visualization of spectral CT studies.

- Provide sufficient features and tools in the visualization application to evaluate the intermixing framework and compare the results with traditional visualization techniques.

With the achievement of these three objectives this research produced a visualization research application called MARSCTExplorer. MARSCTExplorer can test, evaluate, and store algorithms for visualizing spectral CT data. In addition, MARSCTExplorer supports traditional visualization techniques including the transfer function to allow comparisons between visualization techniques and also between spectral and non-spectral CT.

1.4 Thesis Outline

Chapter 2 covers all theory and research contributing to this thesis. This includes the MARS scanner technology, current volume visualization tech-

niques, spectral CT and similar data type characterizations, and a discussion of strategies for visualizing spectral CT data.

Chapter 3 describes the development of an intermixing framework for visualizing spectral CT data. This includes the framework goals and three design iterations together with tests performed to evaluate each design.

Chapter 4 is an overview of the visualization application *MARSCTExplorer*. This includes the graphical user interface, the internal structure, the toolset provided, and the implementation of the intermixing framework from Chapter 3.

Chapter 5 evaluates MARSCTExplorer and the intermixing framework. Four spectral CT studies containing unique materials are visualized using a variety of algorithms. The results show the power of the intermixing framework, the advantages over traditional visualization, and the performance achieved with MARSCTExplorer.

Lastly, Chapter 6 summarizes the designs and results achieved for the intermixing framework and MARSCTExplorer. Some important shortcomings are mentioned with potential solutions and potential future endeavours are discussed.

Chapter II

Related Works

This chapter covers all related theory and works which contribute to this research. This starts with CT technology to describe the current MARS spectral CT prototype. This is followed by a description of volume visualization and a characterization of spectral CT data. Lastly, this is brought together in a discussion of potential visualization techniques in the context of techniques successfully applied to similar data types.

2.1 Computed Tomography Acquisition

The invention of CT has been attributed to Sir Godfrey Hounsfield [5]. Then called computerised transverse axial tomography, Hounsfield managed to quantify attenuation values per voxel by using multiple x-ray projection images. This was the basis for modern CT.

At the core of CT technology is radiography. A source sends x-rays through an object to measure x-ray attenuation via a detector on the other side. CT extends this by acquiring many radiographical images at known orientations. Since the first scanner by Hounsfield, seven generations of CT scanners have evolved [6].

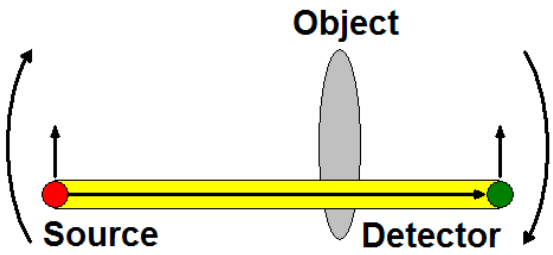

Figure 2.1: The setup for a pencil beam scanner.

2.1.1 Generations of Computed Tomography Scanners

Each generation of CT represents a significant change to the hardware. These changes reduced both errors and acquisition time. The generations of CT scanners are summarized as follows. For more details along with a comprehensive overview of CT, see Bushberg [6].

The original generation was called the *rotate/translate* or *pencil beam* scanners as seen in Fig. 2.1 inspired by Bushberg [6]. This system has a single detector thereby limiting the useful beam to a thin pencil which corresponds to a single pixel in a projection image. Translation constructs the full projection image, and rotation moves the gantry (structure containing the source and detector) in position for the next projection image.

The first generation suffered from high acquisition times which were well over a minute per slice. On the other hand, x-ray scattering was not an issue as only one pixel was acquired at a time. Still this also meant that the exposure needed to be excessively high to produce readable measurements thereby resulting in a high dose.

The second generation increased the number of detectors along the acquisition slice. This was called the *narrow fan beam* scanner as seen in Fig. 2.2.

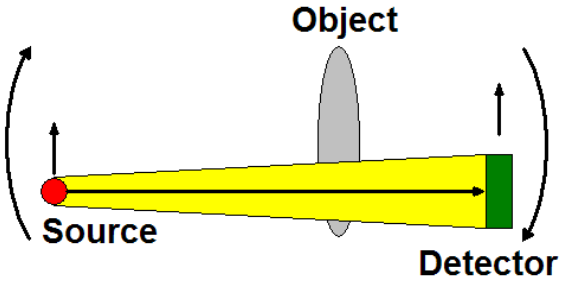

Figure 2.2: The setup for a narrow fan beam scanner.

The use of multiple detectors greatly reduced the acquisition time. However, the whole patient width was not captured so translation was still required. In addition, the use of multiple detectors resulted in the acquisition of scattered photons which introduced additional errors into the image.

The third generation extended the detector row until the whole width of the patient could be captured. This was called the *rotate/rotate* or *wide fan beam* scanner as seen in Fig. 2.3. This reduced potential mechanical error by removing detector translation while also further reducing acquisition time.

A major issue with the first three generations of CT scanners was the ring artifact demonstrated in Fig. 2.4. If a detector error manifests in the same location in all projection images then a ring is produced in the reconstructed

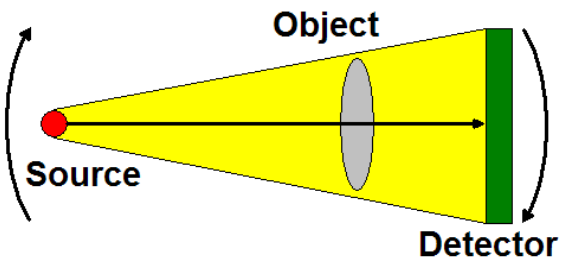

Figure 2.3: The setup for a wide fan beam scanner.

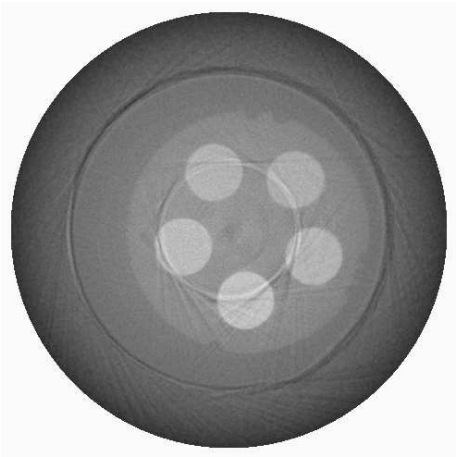

Figure 2.4: Ring artifacts are errors which are present in all or most projection images. These manifest as solid rings in the reconstructed slices.

data which can reduce the diagnostic value of the final image. While it is possible to account for the detector errors, this requires intimate knowledge about each detector which is time consuming to collect.

The fourth generation partially solved the issue of ring artifacts. Here the detector row was extended to a full ring around the slice. This was called a *rotate/stationary* scanner as seen in Fig. 2.5. The importance of this was two-fold. Firstly, the detector elements are stationary allowing exact

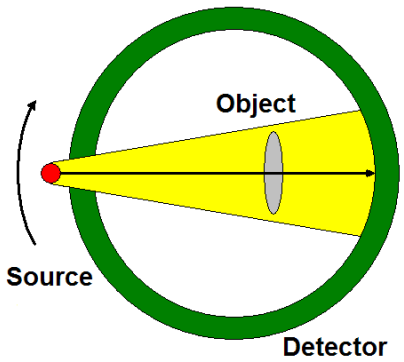

Figure 2.5: The setup for a rotate/stationary scanner.

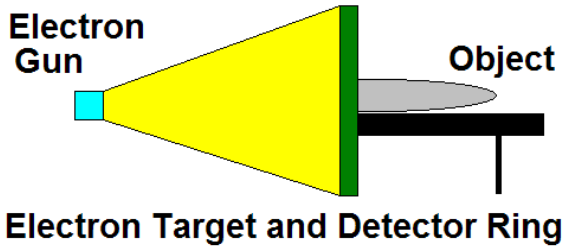

Figure 2.6: The setup for an electron gun scanner.

knowledge of their positions. Secondly, detector errors no longer manifest in the same image location which blurs the effects of ring artifacts over the image.

Moving components are a major hindrance to the acquisition time. The fifth generation was a novel approach yet unpopular due to its size and cost. This was called the *stationary/stationary* or *electron gun* scanner as seen in Fig. 2.6. Here the x-ray source was a ring which was controlled by an electron gun. This meant that there were no physical moving parts and the acquisition time was around 50ms per scan.

The sixth generation is called the *helical* scanner as shown in Fig. 2.7. In this case the patient is translated continuously during acquisition resulting in

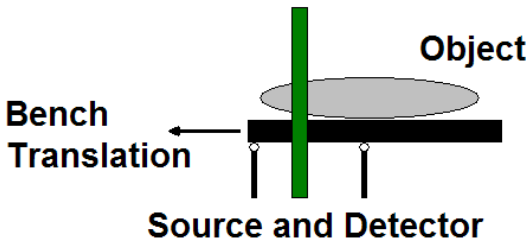

Figure 2.7: The setup for a helical CT scanner.

a helical path for the scan. This reduced the total acquisition time but most importantly allowed the reconstruction of thinner slices using interpolation with no additional scanning time.

The current generation of CT scanners is called the *cone beam* scanner as shown in Fig. 2.8. This has multiple detector rows so that multiple slices can be acquired at a time. The main benefit of this is that the width of the slices can be selected by activating or deactivating detector components. This provided additional control over the quality and exposure required per scan. Other consequences of the cone beam scanner include an increase in scattering effects and a further reduction in acquisition times. In fact, modern CT scanners are capable of scanning a head in under 2 seconds.

2.2 Spectral CT Acquisition

Spectral CT acquisition extends existing scanner technology by acquiring more information with the same acquisition time. The acquisition has two major components; the design of the scanner and the energy selection process. Both components significantly affect the resulting data.

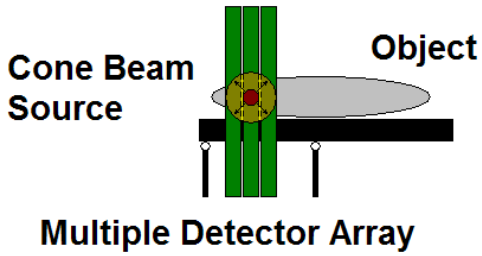

Figure 2.8: The setup for a cone beam scanner.

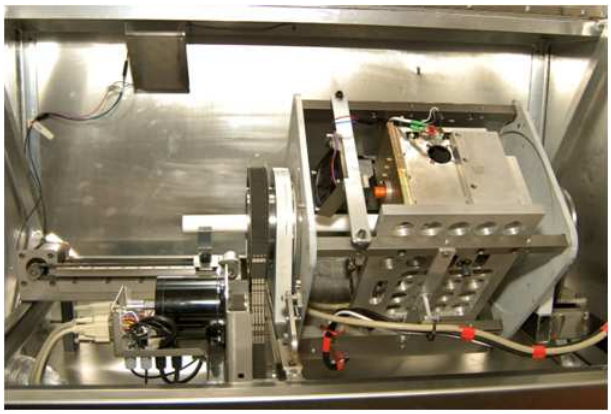

Figure 2.9: The MARS Scanner design. The gantry is visible here together with the x-ray source.

2.2.1 MARS Scanner

The MARS scanner is a purpose built spectral CT scanner designed for specimen and small animal imaging. It is a typical CT design with a gantry that rotates an x-ray tube and detector $0 \rightarrow 360°$ around a target [7]. The gantry of a MARS scanner design is shown in Fig. 2.9.

The detector system uses chips from the Medipix family of energy resolving, photon counting detectors. The Medipix series, developed by CERN [8], is a detector in which the incident x-ray photon interacts with a semiconductor sensor layer to produce a charge cloud which varies with the incident photon energy. This sensor layer is bonded onto an application-specific integrated circuit (ASIC) which analyses the charge cloud for each photon independently in the x-ray beam and records the charge produced, giving a measure of each individual photon's energy. Thus, for an x-ray beam containing many photons, the spectrum of x-ray energies is recorded.

The Medipix family includes a variety of detectors. Medipix2-MXR, Timepix and Medipix3 are the detectors used for the spectral CT studies

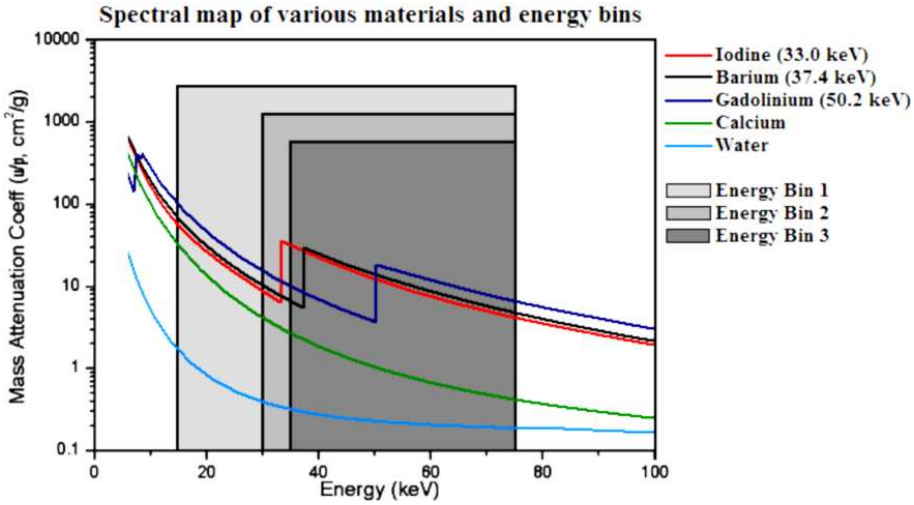

Figure 2.10: Material attenuation over the x-ray spectrum.

in Chapter 5. Each detector can be operated in several modes, but typically 1 to 8 "low thresholds" were used resulting in overlapping spectra as shown in Fig. 2.10. Depending on the detector, thresholds can be obtained either as a series of exposures with a few thresholds (Medipix2 and Medipix3) or as a single exposure with multiple thresholds (Medipix3 only).

The Medipix detectors are 14 × 14mm in size containing a matrix of 256 × 256 independent pixels where each pixel is 55μm^2. To scan larger objects, the detector can be translated for each projection angle to ensure complete coverage of the subject. The current scanner covers a volume of 80mm diameter and 14mm length. Longer objects are obtained by translation of the object between rotations. Typical data sets aquired by the current MARS prototype and used for this research contain up to 1536×512 pixels per projection image (approximately 80mm diameter and 28mm long). These projection images are reconstructed into 3D volumes via traditional filtered back-projection using Octopus [9].

2.2.2 Energy Bin Selection

Spectral CT imaging involves greater complexity during pre-scan planning, as the energy selection process needs to target appropriate x-ray spectra.

The benefit of spectral CT imaging is the ability to target materials. In general it is expected that the important materials in the subject are known. If not then the best option is to perform a study with constant threshold intervals. This would provide the best spread of information to produce an energy bin selection for future studies.

When the important materials are known the first step is to research their attenuation properties. Fig. 2.10 demonstrates this information as a graph showing the mass attenuation coefficient versus energy. Other graphs may include the linear attenuation coefficient or the scaling unit after reconstruction, for example, Hounsfield units.

The spectral characteristics of the target materials shown in Fig. 2.10 can be used to aid energy selection. For example, barium and iodine are similar for the whole x-ray spectrum except between their K-edges at 33.4 and 37.5keV respectively. Therefore the contrast between iodine and barium can be maximized if this spectral region is targeted. This is called K-edge imaging.

The selection process has a few important consequences. Firstly the selection directly affects the contrast which is available for materials of interest. Secondly, the selection process can provide clues as to how the information should be extracted during visualization. Ideally, visualization will be considered during the selection process. Lastly, if an energy bin selection proves to be successful (the visualization produced the desired results), then the

selection can be recorded and reused as a preset for similar studies.

2.3 Visualization of computed tomography

This section presents some of the visualization approaches used with CT. The traditional approach for visualizing CT is to view 2D slices. However, 3D volumetric visualization is steadily becoming the norm. There is a variety of common 3D rendering algorithms for volumetric data. These include intensity projection, isosurfacing, and direct volume rendering. This section provides a summary of these algorithms but for a more in-depth overview of state of the art visualization approaches, see [10] [11].

2.3.1 2D Approach to Volumetric Data

The 2D approach to visualizing CT data was the dominant approach for over 30 years. The reason for this was two-fold. Firstly, computer hardware could not support volumetric algorithms until recent years. Secondly the 2D approach was a natural extension of the traditional x-ray which meant that radiologists could easily adopt 2D visualization software.

2D visualization software is well developed with a number of standard features. Typically, the software will show a variety of views including standard orientations such as transverse, saggital, and coronal views, a view containing key slices to provide an overview of the volume, and a view showing an orientation or scout image. In addition, the views are usually synchronized to allow instant updates between views. A typical screenshot of a 2D visualization application is shown in Fig. 2.11.

For a large portion of CT volumes, the dynamic range of the data ex-

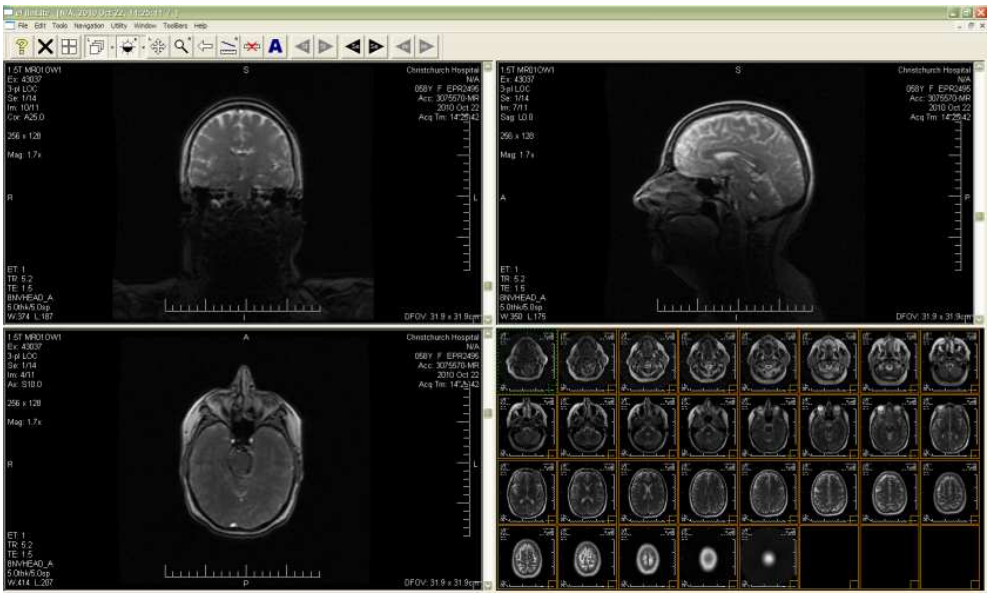

Figure 2.11: Typical 2D visualization software uses a variety of standardized views to allow fast navigation of volumes. Image courtesy of eFilmLite.

ceeds 8-bit. This means that the data can not be completely visible on a typical screen which is restricted to 8-bit. Therefore, 2D visualization software provides a useful tool called the windowing tool. This tool applies a ramp function to the data as shown in Fig. 2.12.

All values in a CT dataset are typically normalized to Hounsfield units. This unit is defined by equation 2.1 where μ and μ_{H_2O} are the attenuation coefficients for the current voxel and water respectively. Many common materials in the human body (e.g. air, fat, bone, water) have sufficiently consistent values that the image can be optimized to display these materials.

$$HU = \frac{\mu - \mu_{H_2O}}{\mu_{H_2O}} \cdot 1000, \tag{2.1}$$

This means that the windowing tool allows the contrast to be maximized for a desired material. As the Hounsfield unit is standardized, the windowing

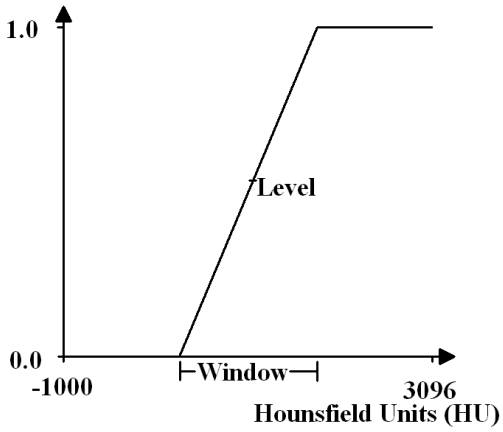

Figure 2.12: Windowing selects a data range and central level. A ramp function is then applied with these settings to maximize contrast for a desired set of values.

tool can support presets for CT data. Unfortunately, this standardization of Hounsfield units is lost with spectral CT data rendering the traditional presets obsolete.

2.3.2 Intensity Projection

Intensity projection is a pseudo 2D technique with two variations. A single value, which is either the maximum or the minimum value, is located

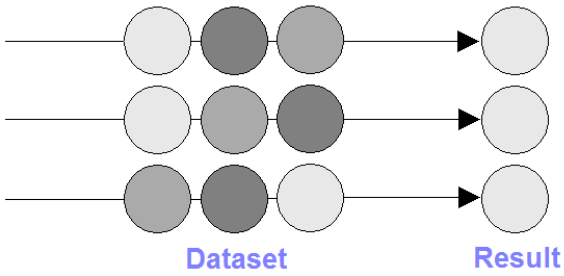

Figure 2.13: Maximum intensity projection extracts the maximum value along a ray. This forms a projection image.

along a ray and projected to the screen. The process for maximum intensity projection is illustrated in Fig. 2.13.

The simplicity of intensity projection meant that implementations were computationally efficient and therefore popular. The issue with intensity projection is that by using only a single value per ray, depth is lost and most of the volume is excluded. Therefore, intensity projection is not a suitable technique for most applications.

On the other hand there are cases where intensity projection is an ideal technique. With maximum intensity projection, high density objects are singled out as shown in Fig. 2.14. This is ideal for thin objects such as blood vessels injected with a high density contrast agent which can easily be obstructed by surrounding objects in other rendering algorithms. Maximum intensity projection is still the best technique to reveal thin objects in a fast and efficient manner.

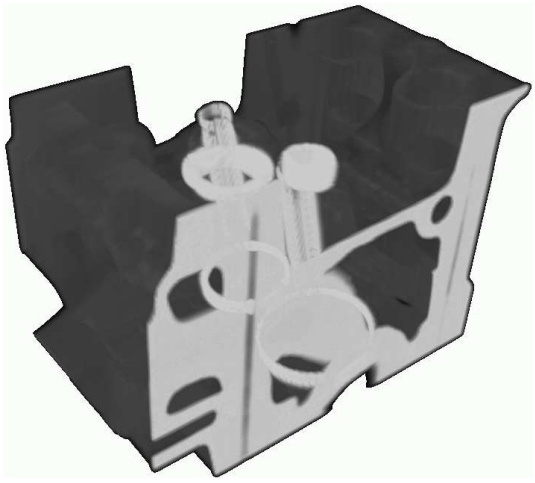

Figure 2.14: Maximum intensity projection of an Engine (dataset courtesy of Volvis.org).

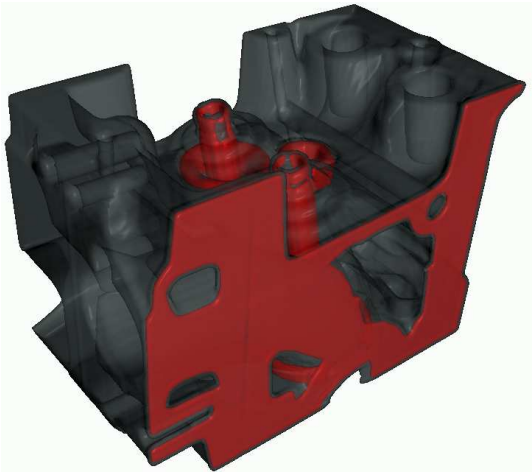

Figure 2.15: Iso-surfacing is an excellent technique to visualize outer borders of materials when the contrast is high.

2.3.3 Iso-surfacing

Iso-surfacing is the technique of visualizing one layer of the volume at a time as shown in Fig. 2.15. As CT volumes define continuous fields, each value corresponds to a surface which can be visualized independently.

There are two common approaches to isoisurfacing. The first method is to extract the surface in a preprocessing step and generate a polygonal model. The second method is real-time iso-surface extraction which is a raycasting-based technique.

The polygonal method commonly adopts the marching cubes algorithm [12]. This method was favored in the past due to the high rendering speed. The basic process is to store various configurations of voxel cubes (illustrated in Fig. 2.16) and compare these to sub volumes in the data. Based on these configurations a polygonal surface is generated.

The main benefit of the polygonal method is the rendering speed. The construction of a polygonal model drastically reduced the data size and also

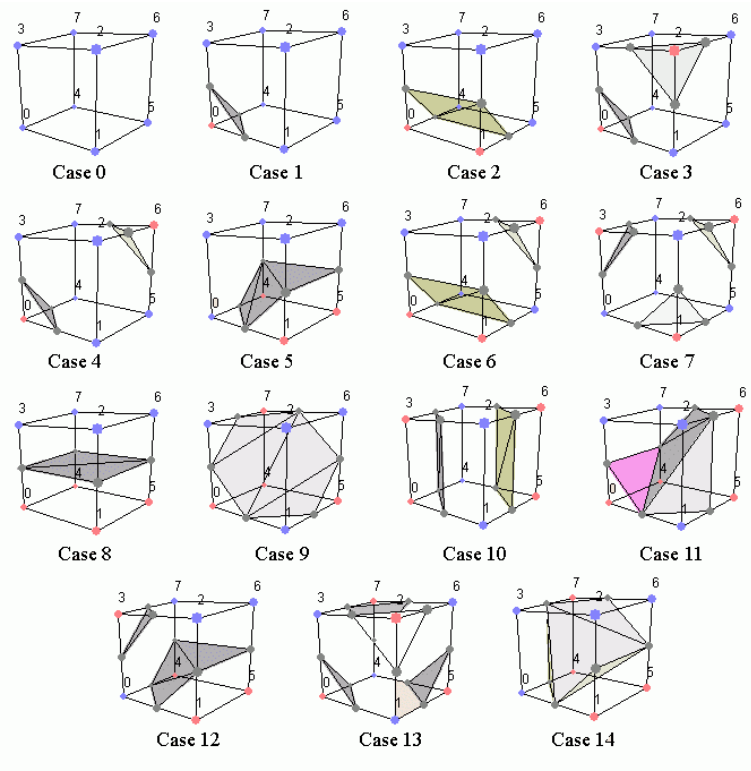

Figure 2.16: Iso-surfaces are built from a finite list of cases, each defining a different type of surface. Courtesy of users.polytech.unice.fr.

made use of the old openGL fixed functionality pipeline which was heavily optimized with hardware acceleration.

Unfortunately, the polygonal method suffers from a few important limitations. Firstly, the method is not interactive as the marching cubes algorithm has to be repeated for each iso value. Secondly, some of the configurations were ambiguous which led to severe artifacts in the polygonal model. Statistical techniques [12] can reduce, but not completely solve this issue. Lastly, the marching cubes algorithm produces more polygons then required. The dual marching cubes algorithm [13] was an improvement which solved this issue.

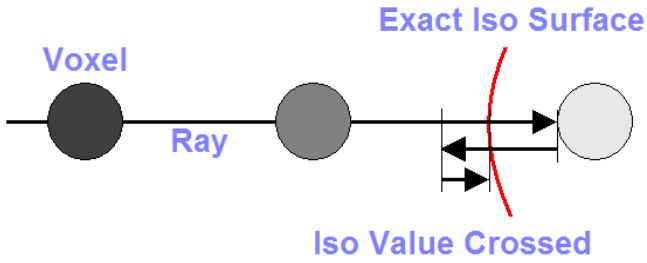

Figure 2.17: Real-time iso-surface extraction casts rays until an iso-value is crossed. The true iso-surface is then found and accumulated.

In recent years, improvements in graphics hardware have allowed real-time iso-surface extraction during rendering. Here the rendering algorithm progresses over a ray through the volume searching for boundaries crossing over the iso-value. The iso-surface is then located and accumulated as necessary. This process is illustrated in Fig. 2.17.

The real-time method has a few important advantages. Firstly, there are no ambiguity issues and hence no artifacts. Secondly, the algorithm is easily adaptable to find multiple iso-values at the same time. In fact the only real issue posed by real-time iso-surfacing is under sampling which can result in thin objects being skipped. This issue can be avoided by maintaining high sampling rates or implementing an adaptive sampling algorithm [10].

2.3.4 Direct Volume Rendering

Direct volume rendering is the most realistic and the most recent class of volume rendering techniques. As the name suggests, the volume is rendered *directly* from the dataset as opposed to a polygonal model. This means that the rendering is computationally intensive. Various rendering algorithms exist to optimize the speed or realism of the result.

Almost all direct volume rendering algorithms treat the volume as a

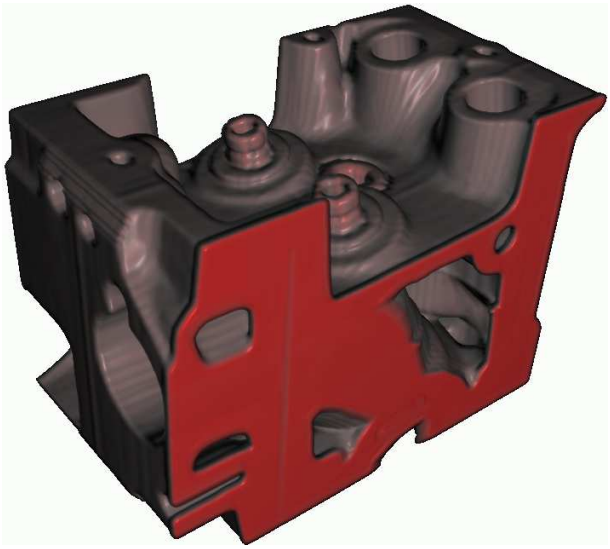

Figure 2.18: Direct volume rendering reveals the whole volume as a gaseous cloud. However, control over the opacity can give a solid appearance.

gaseous material. This allows lighting effects (such as absorption, emmission, and scattering) to be simulated throughout the entire volume. A typical result for direct volume rendering is shown in Fig. 2.18.

The current rendering algorithms include 2D and 3D texturing, splatting, shear warping, and raycasting [10]. Of all rendering algorithms, raycasting is the most natural and produces the highest quality results. Although raycasting has higher computational requirements, the algorithm can easily exploit modern hardware acceleration techniques allowing interactive frame rates in modern systems [11]. Therefore, raycasting is quickly becoming the preferred rendering algorithm for modern medical imaging applications.

Raycasting, as illustrated in Fig. 2.19, sends rays through the volume and accumulates lighting effects along the way. This is a complex process requiring a few key steps including the ray setup, interpolation, sampling, illumination, shading, and accumulation.

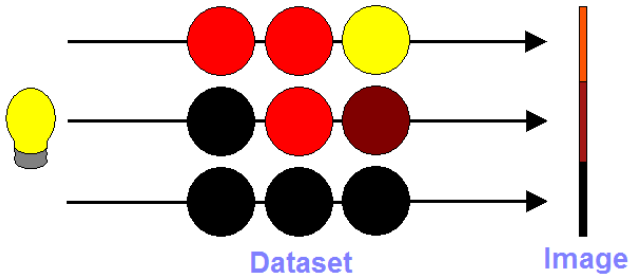

Figure 2.19: Raycasting sends light rays through the volume and accumulates samples along the way.

The *ray setup* is the first step of the raycasting algorithm which finds the starting and ending positions of the ray. The modern method uses hardware accelerated rasterization [11] of a bounding polygonal model as shown in Fig. 2.20. This allows empty space to be excluded.

Raycasting samples the data at various positions along the ray. This means that given a discrete dataset, some form of *interpolation* is required to retrieve the correct sample value. Errors in interpolation results in visual artifacts including pixelation and wood grain artifacts as demonstrated in

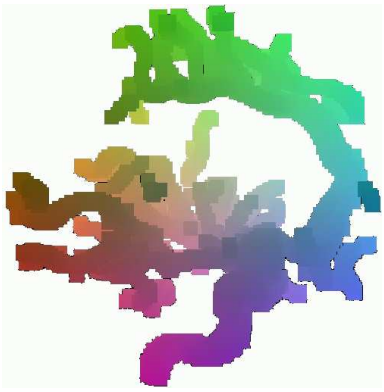

Figure 2.20: Hardware accelerated rasterization uses a polygonal geometry to find the start and end of each ray. The geometry is commonly a set of bricks. Image courtesy of Westfälische Wilhelms-Universität Münster.

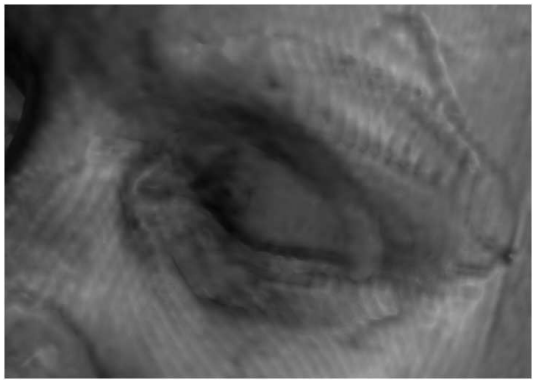

Figure 2.21: The most common volume rendering artifact is the wood grain artifact. This is caused by errors in interpolation schemes and/or undersampling.

Fig. 2.21.

Modern graphics hardware supports two hardware accelerated interpolation schemes; nearest neighbor and linear (trilinear for volumes) filtering. Nearest neighbor filtering clamps to the closest value while trilinear approximates the value via linear interpolation. Better schemes such as tricubic interpolation exist at the cost of severly reduced rendering speed. The common interpolation schemes are illustrated in Fig. 2.22.

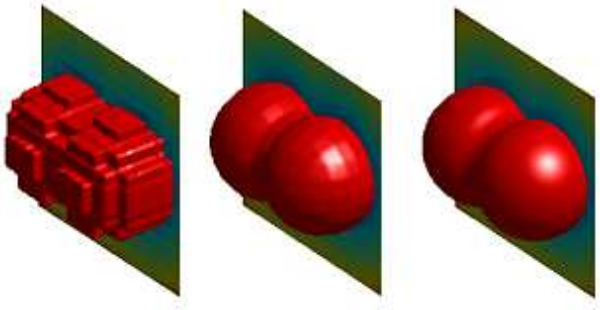

Figure 2.22: The difference between interpolation schemes is clearly visible. Left: nearest neighbour, Middle: trilinear, Right: tricubic. Image courtesy of www.mathworks.com.

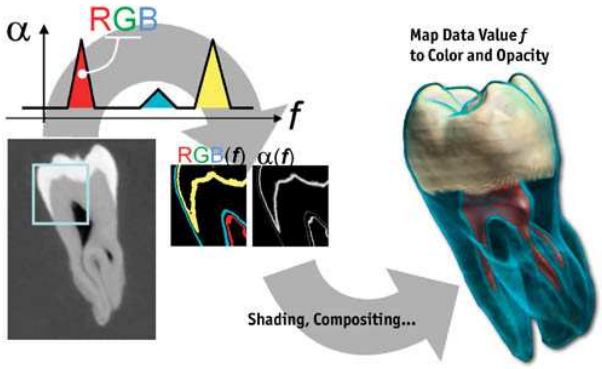

Figure 2.23: Transfer functions are colour maps from raw data to illumination parameters such as red, green, blue, and opacity. Image courtesy of GPU Gems Chapter 39.

The *sampling* is the first step in the rendering loop over the ray. At the sampling stage the raw data is extracted from the volume. At this point, more empty space can be skipped, by thresholding raw data values that are not required. In this way, the sampling steps determine the valid volumetric regions, and hence, the geometry which is present during visualization.

Illumination is the process of converting the raw data into illumination parameters. Typically this will be colours such as red, green, blue and an opacity value (RGBA). The traditional method uses a transfer function as shown in Fig. 2.23. A transfer function is a colour map which is stored as a look up table.

The transfer function has many distinct benefits. Firstly as a lookup table it is also possible to preprocess part of the accumulation step, thereby simplifying the rendering process and also reducing visual artifacts [14]. Secondly, as CT data usually represents Hounsfield units, the transfer function can also serve as a preset. However, as mentioned before, Hounsfield units change with energy preventing the use of transfer functions as presets for

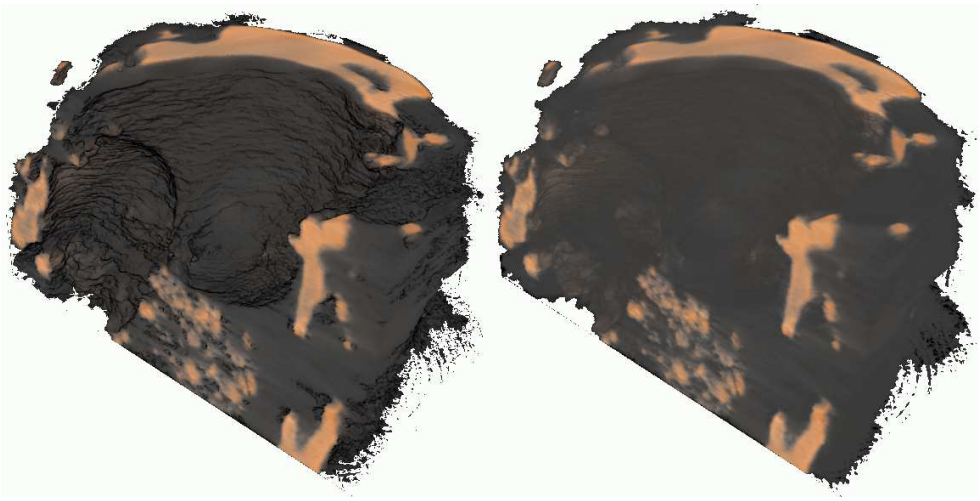

Figure 2.24: Shading provides important surface details and visual cues essential to a good understanding of the volume geometry. The lumen geometry of the atheroma plaque dataset is clearly visible in the left image with shading but is not so clear in the right image without shading.

spectral CT data.

Shading simulates the effects of external lights. Typically this is based on the local gradient in the volume which is also the normal of the iso surface at the sample. This method allows an approximation of the lighting effects using traditional models such as the Blinn-Phong model. The effect of shading is important as it provides most of the surface detail. Fig. 2.24 compares direct volume rendering with and without shading.

The final step of raycasting is the *accumulation* of the samples. For raycasting the accumulation step integrates the volume rendering integral shown in equation 2.2 which represents absorption and emission of light along a ray [10].

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)} d\tilde{s} \tag{2.2}$$

I is the intensity, s is the position, τ is the transparency and q is the emission. This is restructured as the compositing equations given in 2.3 and 2.4 for iterative computation.

$$C_i^{'} = C_{i+1}^{'} + \left(1 - A_{i+1}^{'}\right) C_i, \tag{2.3}$$

$$A_i^{'} = A_{i+1}^{'} + \left(1 - A_{i+1}^{'}\right) A_i \tag{2.4}$$

C is the colour value and A is the opacity (1.0 - τ). Colours such as red, green, and blue can be independently computed.

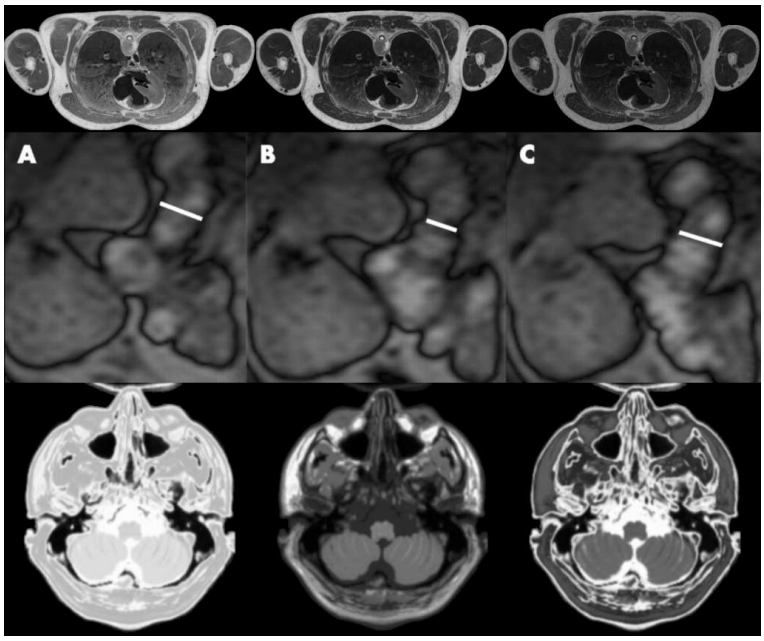

Figure 2.25: Three multi-variate data types similar to spectral CT data. Top row: three multi-channel datasets, courtesy of the visible human project. Middle row: three time-varying datasets, courtesy of gut.bmj.com. Bottom row: three multi-modal datasets, courtesy of volvis.org.

2.4 Multi-Variate Data

In recent years medical imaging has produced novel data types, in particular, multi-variate data types. Multi-variate refers to multiple variations or multiple datasets that describe a single object. Of the various multi-variate data types, three have similarities to spectral CT data. These are the *multi-channel*, *time-varying*, and *multi-modal* data types. Fig. 2.25 shows examples of these three data types with different datasets for a single slice.

In addition, there are also different data representations including *scalar* datasets, *vector* datasets and *tensor* datasets. Fig. 2.26 shows an example of each data representation. For a comprehensive review of data types, data representations, and common visualization techniques thereof, see [15].

2.4.1 Multi-Channel Data

The most prominent example of a *multi-channel* dataset is the visible human project [16]. Here a human cadaver of a prisoner was cut in fine slices with a laser (with explicit permission before death) and photos were taken of each slice. The obtained volume was therefore in RGB format where the red,

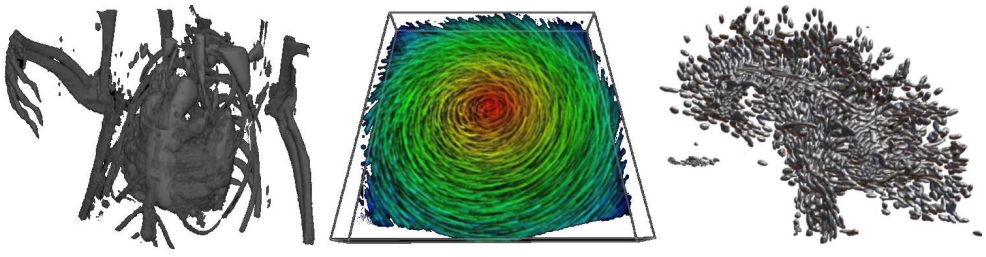

Figure 2.26: Three typical examples of volume rendering for scalar, vector, or tensor datasets. Left: a scalar dataset. Middle: a vector field. Image courtesy of persons.unik.no/andershe. Right: a tensor field. Image courtesy of www.cs.utah.edu.

Figure 2.27: A section of the visible human male with a 1-1 map for each colour channel.

green, and blue datasets are the individual channels.

Visualization of Multi-Channel Data

Visualization of *multi-channel* data is the most straightforward of all multivariate data types. As the data is already in a format suitable for the volume rendering equation, a simple 1-1 map between colours will suffice as shown in Fig. 2.27. However, this will not provide much control over the volume.

Recent investigations into *multi-channel* visualization turn to alternative colour schemes for better control. In Ghosh's paper [17], the YUV colour scheme is adopted in order to gain better control over different components in the volume. The L channel controls bright areas such as skin and bone while the U controls changes in redness such as bone to muscle.

While Ghosh's technique is not immediately translatable to spectral CT, his use of different colour schemes has great benefits. In the spectral CT sense, the data represents attenuation or Hounsfield units and both are a loose measure of density. Therefore, it can be expected that a colour scheme such as hue, saturation, value, and alpha (HSVA) can be used to define

colours based on density.

2.4.2 Time-Varying Data

As the name suggests, *time-varying* data contains multiple datasets representing time dependent key frames. *Time-varying* data can be extremely useful for functional based imaging in the medical field. Measurements of blood flow and correct heart beat function can be determined from time-varying data.

Visualization of Time-Varying Data

As with multi-channel data, the visualization of *time-varying* data generally uses animation as the natural visualization technique. However, this can have complications as volume rendering is slow by nature. In addition, animation is heavily dependent on significant changes which can capture the users attention. Subtle events can easily be missed.

To solve this Woodring developed an intermixing framework for *time-varying* data [1] where intermixing is the combination of data to produce a single visual outcome. Intermixing has also been referred to as comparative visualization.

Woodring's intermixing framework is both interactive and generic. The framework is based on a set of eight operators derived from Cai's constructive volume geometry work [4]. With these operators the keyframes can be combined in simple steps to construct a volume tree as in Fig. 2.28.

The greatest benefit of the volume tree is the ability to visualize, and edit each node. This allows the user to progress through a complex algorithm step

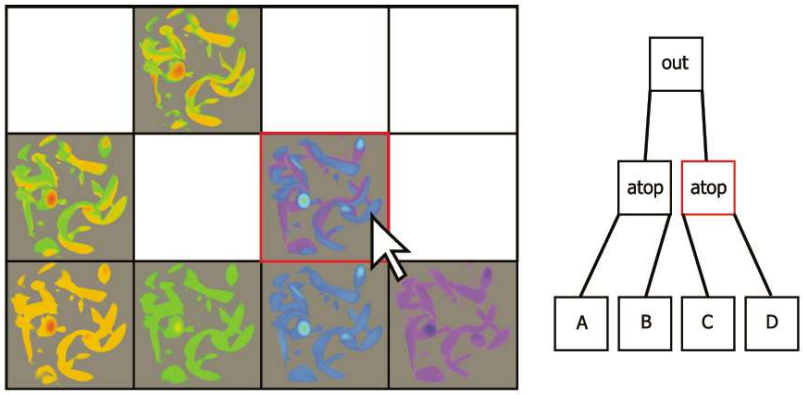

Figure 2.28: The volume tree from Woodring's intermixing framework [1] allows each step to visualized and traced to the raw data.

by step to fully understand the effects of each part of the algorithm. Based on the algorithm selection, Woodring's framework allows even subtle changes in time-varying data to be indentified and visualized

2.4.3 Multi-Modal Data

Multi-modal data occurs when each dataset is independently acquired. This includes any medical imaging modality such as MRI (and variations of MRI), CT, PET, SPECT, or ultrasound. The motivation for such data is that each modality provides unique information not supplied by the others (MRI is good with fluids, CT is good with dense materials).

The initial use of *multi-modal* data is described in [18]. The biggest issue with *multi-modal* data is the fact that the subject will almost never have an identical geometry between modalities. Hence, *multi-modal* data requires an additional preprocessing step called *registration* (also called spatial registration).

Registration is the process whereby multiple volumes are distorted, trans-

Figure 2.29: A PET-CT scanner which acquires CT and PET datasets simultaneously to bypass registration requirements. Image courtesy of www.ccsb.org.

lated, and rotated to align with each other. This is a difficult and time-consuming process with no guarantee of complete alignment.

In more recent years, medical imaging hardware has been designed specifically for *multi-modal* acquisition. These designs combine modalities into one device such as PET-CT shown in Fig. 2.29. A single device means that either no registration is required or perfect registration can be achieved from the known geometry. The MARS scanner design is such that no registration is required.

Visualization of Multi-Modal Data

The visualization of registered multi-modal data is a complex field due to the variance in the physical data unit of each mode. In some cases it is straightforward, as in PET-CT, where the activity described by PET is most effective as a colour scheme while the CT functions well as a geometric de-

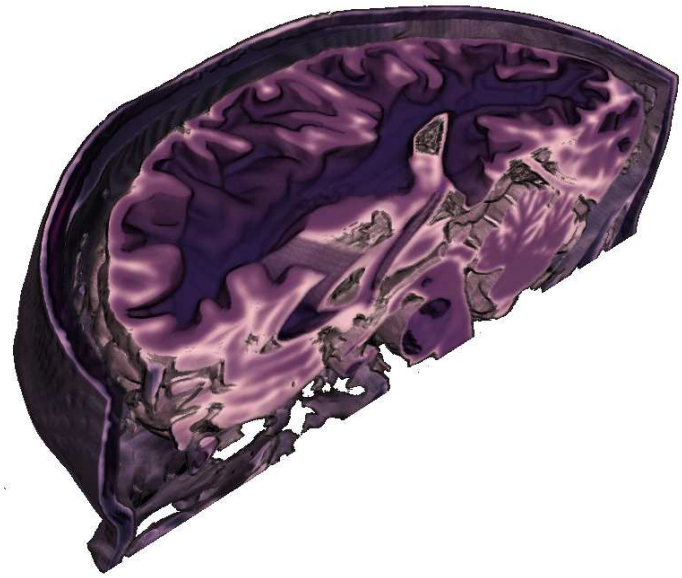

Figure 2.30: Multi-modal visualization can vary significantly as each mode represents different physical phenomena. However, intermixing still forms the core of most multi-modal visualization techniques. This is an MRI head dataset with PD, T2, and T1 datasets assigned to an HSVA illumination model respectively.

scriptor. However, intermixing is the core of most multi-modal visualization techniques. Other visualization techniques include temporal (one mode at a time) and spatial (side-by-side) visualization. A typical result of multi-modal visualization is shown in Fig. 2.30 with an MRI dataset.

A complete overview of techniques used in multi-modal visualization is described by Ferre [19]. In his article, Ferre refers to different classes of intermixing algorithms as fusions including property fusion, material fusion, gradient fusion and others. In addition, for each fusion, Ferre considers two intermixing algorithm types, namely, one property per point or multiple properties per point.

The main contribution from Ferre's paper is the intorduction of a generic rendering pipeline model for each fusion and property per point configuration.

Ferre demonstrates how each modality may be considered independently and in a structured fashion to produce a single, combined image. If a modality has additional information about the material of a voxel then, naturally, material fusion is required. In this way the large variation in data units over the various modalities is taken into account and can be visualized by a single application.

2.5 Spectral CT Data

Spectral CT data is a single volume described by multiple energy bins. Therefore, spectral CT data is a multi-variate data type. Fig. 2.31 shows an example slice with three energy bins acquired with the MARS scanner. The maximum energy bins for a single spectral CT study acquired to date is 24 energy bins although most studies typically contain 4 to 8 energy bins.

The data is currently normalized to the Hounsfield scale which provides some measure of standardization. However, the Hounsfield numbers can no longer be treated as fixed CT numbers as the Hounsfield scale changes with energy.

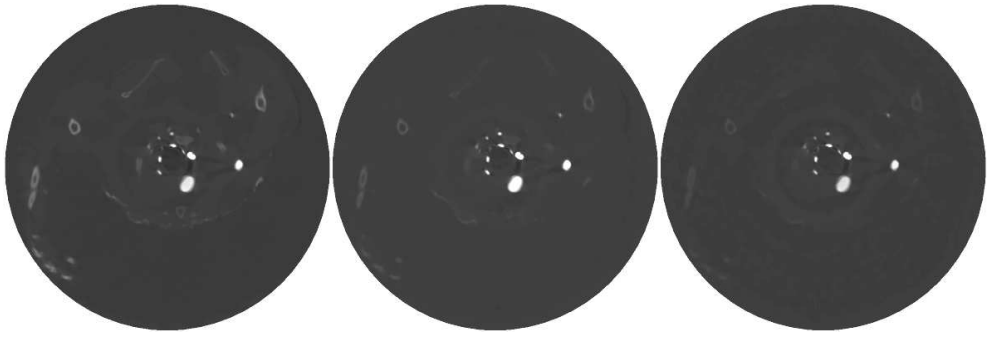

Figure 2.31: A single slice from a spectral CT dataset showing three energy bins at 15keV (left), 30kev (middle) and 35keV (right).

2.5.1 Potential Techniques for Visualizing Spectral CT Data

Spectral CT data is a novel data type. This means that there is currently no prior research into visualizing spectral CT data. The starting point of this research was the working theory that visualization techniques successfully applied to similar data types, should also be successful for spectral CT data.

Each energy bin provides unique information about both the geometry and material content of the volume. This is similar to multi-modal data where each modality is selected based on the unique information it provides.

Each energy bin represents the x-ray attenuation of materials over a unique x-ray spectrum. Multi-channel data represents colours which in turn represent unique spectra from the visible range. In the same way, the energy bins can be thought of as representing unique colours.

Lastly, each energy bin forms part of a continuous curve over the x-ray spectrum. This continuity is similar to that with time for time-varying data.

In the previous sections, a few important visualization techniques were identified for the multi-modal, multi-channel and time-varying data types. These are temporal, and spatial visualization, intermixing, and animation.

Both temporal and spatial visualization are simple techniques. In both cases, the visualization application simply requires multiple views. Temporal visualization switches between views while spatial visualization shows these views at the same time.

The downside of temporal and spatial visualization is that big differences are required between datasets to enable the user to compare energy bins. Spectral CT often has subtle differences for the important materials. Therefore, both temporal and spatial visualization are not ideal but are easily

implemented.

Animation is a visualization technique used exclusively for time-varying data (note that this does not include the creation of animations for video output). The use of animation with spectral CT is expected to produce some useful results.

As seen before in Fig. 2.10, K-edges provide distinct jumps in the x-ray spectrum which are unique to each material. An animation with energy as the time variable should reveal K-edges at distinct frames over the animation. Unfortunately, a large number of energy bins would be required to accurately reconstruct the spectral curves in Fig. 2.10. Therefore, animation is not expected to be a useful option for the current spectral CT prototypes.

Intermixing is a visualization technique with successfully applied to all three similar data types. Therefore, intermixing is expected to have the best success with spectral CT data.

2.6 Miscellaneous Works Contributing to this Research

There is other useful research which is relevant to spectral CT visualization. High dynamic range volume rendering [20] maintains the precision provided by datasets. Spectral volume rendering [21] adopts a spectral illumination technique during rendering. While these techniques were not implemented, they could be useful additions to spectral CT visualization.

2.6.1 Prior Work: Data Filtering

Prior to the research performed in this thesis, some preliminary work was conducted. The spectral CT data acquired by the current MARS scanner

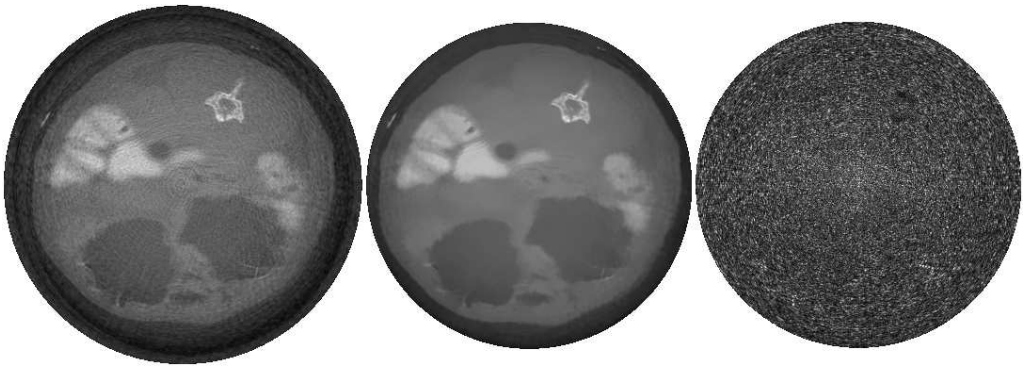

Figure 2.32: A typical result from the CUDA accelerated non local means filter. The original image is on the left, the filtered image is in the middle and the difference image is on the right. Ideally, the difference image should be pure noise.

prototype often has ring artifacts and salt and pepper noise. To guarantee a high level of quality in this research, a CUDA accelerated non-local means (NLM) filter [22] was developed.

The NLM filter is a slow but high quality denoising filter which preserves boundaries such that there is no visible loss of data. The application produced uses CUDA technology to accelerate the process and has been published in [14]. Fig. 2.32 shows the results achieved with the NLM filter.

2.7 Summary of Related Works

The MARS scanner is the world's first spectral CT scanner capable of acquiring any number of energy bins. The current prototype is based on a typical Micro CT design using a rotating source-detector gantry and a cone beam.

The resulting spectral CT data is a novel multi-variate data type with similarities to multi-modal, multi-channel, and time-varying data. This research began with the theory that visualization techniques successful with

similar data types, should also be successful for visualizing spectral CT data. Common visualization techniques identified for similar data types include temporal and spatial visualization, intermixing, and animation. Of these, intermixing was identified as the visualization technique with the best potential.

Chapter III

Intermixing Framework for Spectral CT Data

This chapter details the development of an intermixing framework for spectral CT data. The chapter first introduces design criteria drawn from the spectral CT data characteristics and the aims of this research. The chapter then continues with the development process to meet the described criteria. Lastly, the final design is described in detail together with tests performed on the intermixing framework to validate correct behaviour.

3.1 Motivation for Intermixing

Before constructing a list of criteria for the intermixing framework it is best to recall what is known about spectral CT data and the research goals.

Spectral CT data is a new multi-variate data type made up of energy bins. Energy bins are CT datasets acquired with different x-ray spectra. This results in additional information about the geometry and material content of the volume. Additional geometric information improves rendering quality while additional material information extends CT potential in accurately identifying or distinguishing materials. Therefore, extracting material information is prioritized in this research as it has more clinical utility.

The aim of this research was to provide the foundation for visualizing spectral CT data. Traditional visualization based on transfer functions is not applicable to spectral CT as N energy bins would require an N-dimensional

transfer function which is impractical. In addition, the resulting transfer function would only be valid for that specific set of energy bins. Therefore, new techniques are required to visualize spectral CT data.

To visualize spectral CT data, similar data types were investigated and intermixing was identified as a potentially successful visualization technique. Therefore, this research aims to design and implement an intermixing framework for spectral CT data. More explicitly, the mission statement for this research is as follows.

Create an intermixing framework which can extract and display the enhanced material information from spectral CT data and provide the basis for extracting geometric information.

3.2 Intermixing Framework Criteria

Intermixing is a general term describing the process of combining data mathematically to produce a single visual outcome. Before an intermixing framework can be designed and implemented for spectral CT data, two important research questions must be addressed.

- When should the data be mixed in the visualization pipeline?
- How should the data be mixed to reach the desired outcomes?

3.2.1 Intermixing and the Visualization Pipeline

The choice of when to combine data in the visualization pipeline is dependent on the rendering speed and the level of user control desired. For example, a preprocessing step has no additional rendering load but the step must be

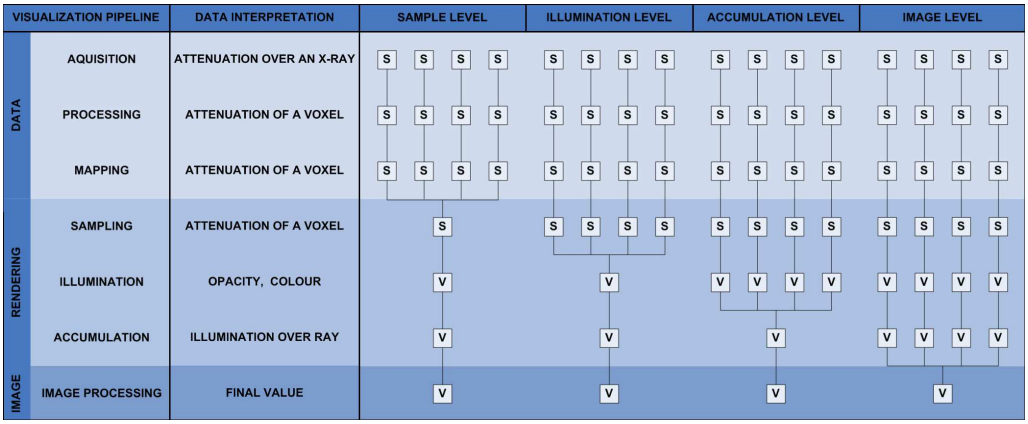

Figure 3.1: Some intermixing strategies over the visualization pipeline together with the physical meaning at each stage. S stands for scalar while V stands for vector.

repeated for every alteration. Alternatively, real-time combination adds an additional rendering load but updates changes instantly.

Another complication is the physical meaning of the data as the nature of the data can change over the various levels in the CT visualization pipeline. This can greatly affect what algorithms are suitable at each level. Based on the visualization pipeline introduced by Bürger [15] and the rendering stages introduced by Cai [4], Fig. 3.1 shows the visualization pipeline used for this research along with the physical meaning of the data at each level.

There are seven levels in the visualization pipeline spread over three stages. The stages split the visualization pipeline based on preprocessing (*Data Stage*), real-time processing (*Rendering Stage*), and the optional real-time postprocessing (*Image Stage*). Intermixing can occur at any of the seven levels over the three stages.

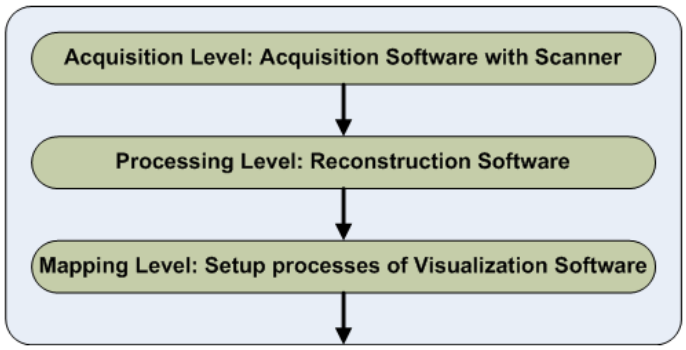

Figure 3.2: The *data stage* has three levels for three different software applications: acquisition software, reconstruction software, and visualization software.

Data Stage

The *data stage* summarizes all processes performed prior to rendering. It contains three levels split by the software which performs the task. This is summarized in Fig. 3.2.

The first level is the *acquisition level* which deals with projection images. Extracting material information from the energy bins would be extremely difficult at this level as each pixel in a projection image represents a multitude of materials over an x-ray.

The second level is the *processing level* which deals with reconstructed volumetric data. Each voxel can be assumed to be sufficiently small to represent a single material. Therefore, extracting material information should be possible at this level.

The third level is the *mapping level*. This level is identical to the processing level except that all processes occur in the visualization software. This is advantageous as it means that fewer processes need to be repeated if a change occurs.

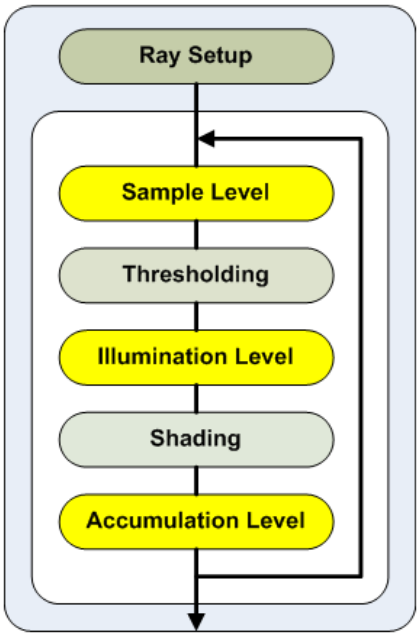

Figure 3.3: This is a simplified rendering pipeline revealing the positions of the sample, illumination, and accumulation levels within the raycasting loop.

Intermixing during the data stages is only beneficial when suitable algorithms are known with no variable parameters. The visualization performance would then be maximized as there is no additional load on the rendering. However, as it is currently unknown what algorithms will be suitable to visualize spectral CT data, recomputing combinations for every alteration would be a serious hindrance. Hence, the *data stage* is not considered for this research.

Rendering Stage

The *rendering stage* includes all processes performed during real-time rendering of the data as illustrated in Fig. 3.3. Of these processes, only three alter the nature of the data.

The first level is the *sample level* which deals with raw volume data. This

level precedes thresholding which means that any processing directly affects what data will be skipped. Therefore, the sample level defines what geometry exists making it a good candidate for extracting geometric information from spectral CT data.

The second level is the *illumination level*. This level translates raw volume data into parameters for an illumination model which is typically red, green, blue, and alpha (RGBA). The colour channels are commonly used to highlight materials while the alpha channel defines the opacity and hence, the visible geometry. Therefore, the illumination level is a good candiate for extracting both material and geometric information from spectral CT data.

It is important to distinguish the roles of the sample level and the alpha channel. The sample level deals with geometric presence while the alpha channel deals with transparency (ranging from 0-1 where 0 is 100% transparent and 1 is 100% opaque). The difference is best understood with transparent solids where the alpha channel will have value 0 while the sample level simply passes the raw data. This is important when shading the transparent solid as only the sample level will pass useable data.

The third level is the *accumulation level* which represents accumulated colours and opacities along a ray. As with the acquisition level, extracting material or geometric information from such data is difficult to achieve.

The benefit of intermixing during the rendering stage is instantaneous updates for all alterations to the intermixing algorithm. As long as interactive frame rates can be achieved, this would be ideal for this research as it is expected that many alterations will occur during visualization.

Image Stage

The *image stage* is an optional stage which combines multiple rendering passes. This can be useful when combining different rendering algorithms such as raycasting with maximum intensity projection. For intermixing spectral CT data, however, this stage has little use and is not considered in this research.

Summary of the Visualization Pipeline

After careful consideration of the visualization pipeline, the data characteristics, and the aims of this research, the first criterion for the intermixing framework is that intermixing will occur at the sample and illumination levels during the rendering stage. The rendering stage supports instant updating of the intermixing algorithms while the sample and illumination levels clearly separate the processes for extracting geometric and material information. These features make the sample and illumination levels ideal for the goals of this research.

3.2.2 Operators for a generic Intermixing Framework

The question of how to combine data is a complex mathematical problem. Any mathematical expression can be used to combine data. However, only a few will produce results which are both meaningful and clear. Therefore, the choice of mathematical expressions to form an intermixing algorithm requires careful forethought, testing, and evaluation.

As this research is the first study performed on visualizing spectral CT, there are no known algorithms for combining spectral CT data. Therefore,

the second criterion for the intermixing framework is support for creating, testing, and storing intermixing algorithms. In this way, the resulting framework can be used to find suitable algorithms for visualizing spectral CT data.

The consequence of the second criterion is that the intermixing framework must have a generic and flexible design. To support algorithm creation the only option is to resort to basic mathematical elements.

Every mathematical expression can be divided into simple steps consisting of an operator and one or two operands. This is summarized by the definition in equation 3.1. Here the operands $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$ can be data samples, constant values, or the results from another operation.

$$\underline{\mathbf{Output}} = Operator(\underline{\mathbf{A}}, \underline{\mathbf{B}}) \tag{3.1}$$

After the generic definition of an operand, the only limiting factor of equation 3.1 are the operators defined. Other intermixing frameworks all use specific operators such as Woodring [1] and Chen's [23] constructive volume geometry operators based on unions and intersections. While useful for combining geometries these operators are insufficient for this research which needs to extract both geometric and material information. Therefore, basic math is the best option to support construction of a wide range of expressions.

Another complication is that the voxels in spectral CT data can be represented in two different ways; as multiple scalar values or as components of a vector.

The vectorial structure is superior for two reasons. Firstly, a vector structure supports the same math as a scalar structure with the addition of linear

Table 3.1: The Operator List

Operator	**Definition**
Addition	$O = A + B$
Subtraction	$O = A - B$
Multiplication	$O = A \times B$
Division	$O = A \div B$
Absolute Value	$O = \|A\|$
And	$O = A\&B$
Or	$O = A\|B$
Exclusive Or	$O = A \hat{} B$
Step Function	$O = \begin{cases} 1.0, for A \geq B \\ 0.0, otherwise \end{cases}$
Maximum	$O = \begin{cases} A, for A \geq B \\ B, otherwise \end{cases}$
Minimum	$O = \begin{cases} A, for A \leq B \\ B, otherwise \end{cases}$
DOT	$O = \underline{A} \cdot \underline{B}$
2 Norm (Length)	$O = \|\underline{A}\|_2$
Pre-Transformation	$\underline{O} = \mathbf{M} \times \underline{A}$
Post-Transformation	$\underline{O} = \underline{A} \times \mathbf{M}$

algebra. Secondly, shader languages, for hardware-accelerated implementations, can vectorize code up to size 4 allowing four operations in parallel. This can greatly improve performance for a complex algorithm.

The resulting operator list is defined in table 3.1. This list consists of fifteen generic operators including five for linear and non-linear expressions, three for boolean expressions, a thresholding operator to create boolean masks, two for basic comparisons, two for basic linear algebra expressions, and two for matrix transformations.

Matrix transformations are useful for data analysis tools such as principal components analysis (PCA) which is applied to data via a transformation. Therefore, the inclusion of transformations as operators allows data analysis tools to be computed during run-time, and applied in real-time.

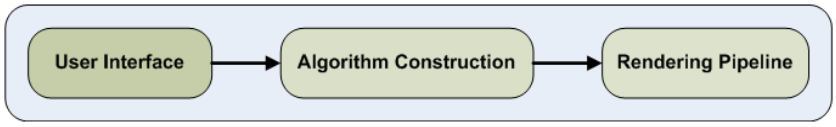

Figure 3.4: A generic intermixing framework has three components. The user interface, the algorithm construction, and the rendering pipeline.

3.2.3 Summary of the Intermixing Framework Criteria

The initial research into spectral CT data, and intermixing resulted in two important criteria for the intermixing framework. The first is that intermixing will operate on spectral CT data during the sample and illumination levels in the rendering stage. The second is the intermixing framework must support the creation, testing, and storing of intermixing algorithms so that suitable algorithms for combining spectral CT data can be found.

These criteria led to the definition of an operator which functions on vectors of size 4. From this definition, 15 operators were defined from elementary math. These operators support the construction of linear, non-linear, and boolean expressions. In addition, matrix transformations support the implementation of data analysis tools such as PCA.

3.3 Development of an Intermixing Framework

The development of the intermixing framework was an interative process over three designs. The first design was a fixed algorithm solution. The second design introduced dynamic shader generation. The final design perfected the system with an object oriented design.

An operator based intermixing framework is built from three components as illustrated in Fig. 3.4. In this chapter, only the algorithm construction

and the rendering pipeline are discussed. The user interface is presented as part of the visualization software MARSCTExplorer which is described in Chapter 4.

The intermixing framework was developed using C++ on a Microsoft Windows platform (Microsoft Visual Studio 2008). All dependencies use open-source libraries chosen to support a cross-platform solution although this was not fully tested. The volume rendering engine uses OpenSceneGraph (an object-oriented OpenGL [24] library). OpenSceneGraph provides a high-level representation of graphical information. In addition, a basic volume rendering module called osgVolume was the sample on which this research was based with extensive modification and re-implementation. The graphical user interface uses wxWidgets (http://www.wxwidgets.org/).

The rendering was implemented using vertex and fragment shaders using GLSL 1.5 [25]. The vertex shader computes the volume boundaries to produce the start and end coordinates while the fragment shader renders the image. The rendering algorithms implemented include raycasting, iso surfacing, maximum intensity projection, and a 2D slice viewer.

3.3.1 Fixed Algorithm Solution

The first design was implemented for a proof of concept. A multitude of shaders were hard coded with a simple intermixing algorithm. This was fast to implement and produced immediate results but did not support algorithm construction.

The shader defined the most basic intermixing algorithm possible. This was a direct map between energy bins and the RGB channels of the illumi-

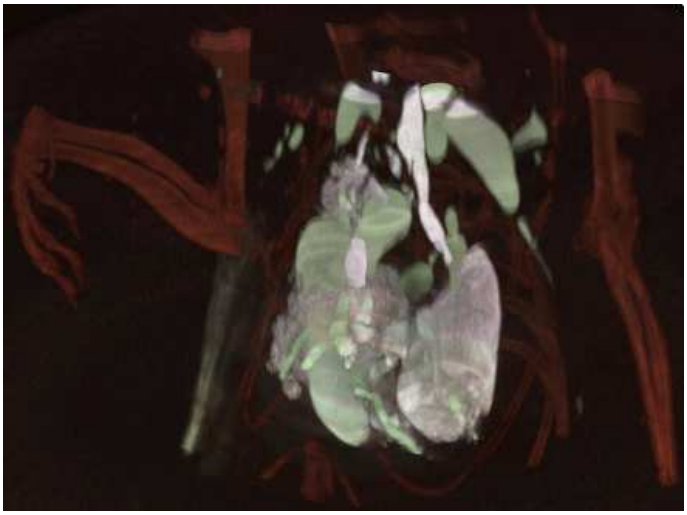

Figure 3.5: The initial results achieved by the fixed algorithm system. Maximum intensity projection was applied to the Mouse12 dataset (see Chapter 5. The 15keV, 30keV, and 35keV energy bins were used in this system.

nation level. Such a map would reveal any differences between the energy bins via distinct colours.

Testing the fixed algorithm solution

The design was tested on the Mouse12 dataset which is introduced in Chapter 5. The initial results of the test is shown in Fig. 3.5.

The goal of the Mouse12 study was to identify three key materials including calcium (bones), iodine (right heart chamber) and barium (esophagus). The initial results in Fig. 3.5 clearly show the three dominant materials as red for calcium, green for iodine, and pale blue for barium.

This result had two important consequences. Firstly, it validated that intermixing can be successful for visualizing spectral CT data. Secondly, it demonstrated that spectral CT imaging can separate iodine and barium which was not possible by non-spectral means. Therefore, this result moti-

vated the continued development of the intermixing framework.

3.3.2 Dynamic Shader Generation Solution

The issue with the first design was a lack of flexibility. Hardcoding algorithms into multiple shaders meant that algorithm construction was not supported. The second design solved this issue by adopting dynamic shader generation.

With this approach, shaders are stored as strings and compiled at run-time before rendering starts. This means that the shaders can be altered during run-time and recompiled to update the rendering process.

The use of dynamic shader generation in volume rendering is not new. Röβler [26] used dynamic shader generation to combine different rendering styles. The rendering styles were defined by code snippets which were pasted into the final shader. This gave the user more control over the visual result while also reducing the amount of shader code which had to written and maintained.

The use of dynamic shader generation in this research was to allow algorithm construction. The idea was to allow users to select data structures and operators thereby creating mathematical expressions. The dynamic shader generation would then translate the expressions into shader code to apply the constructed algorithm.

The goal of this design was to implement the operator system and verify correct behaviour and acceptable performance. To reduce development time, the system adopted a scalar approach, implementing only the first 11 operators from Table. 3.1.

The system of the second design is illustrated in Fig. 3.6. The algorithm

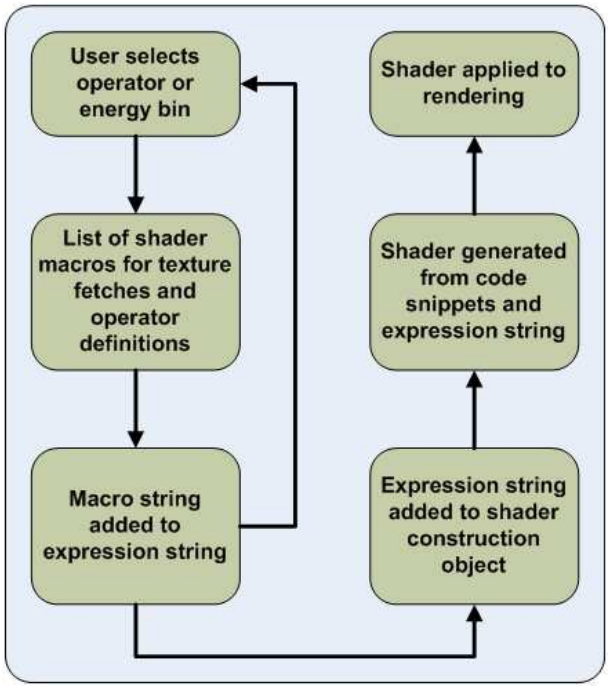

Figure 3.6: The system structure of the second design. This design utilized dynamic shader generation to implement the scalar operators.

is constructed using an "'expression string"'. Each time the user selects an operator or an energy bin, an appropriate string is truncated onto the expression. Upon completion of the expression, the string is passed to a shader construction object called MARSShader. MARSShader combines the expression string with predefined code snippets.

An issue with the operators is the lack of consistency between their implementations in GLSL. For example, addition has the form A + B but the maximum operator has the form max(A,B). This issue was addressed by exploiting macros for the GLSL preprocessor. Preprocessor code are hints to the compiler processed before compilation. In this case, macros can translate the various operator implementations into a standard form as in equation 3.2.

$$\#define \qquad ADD(A, B) \qquad (A + B) \tag{3.2}$$

The benefit of a standard form is that the expression string construction is simplified. In particular, the order by which the user must input operators and operands is standardized. The result is a consistent method to construct algorithms using the first 11 operators.

Testing the Dynamic Shader Generation Solution

The tests performed on the second design were aimed at verifying behaviour and performance.

The first test was to apply each operator on a synthetic spectral CT

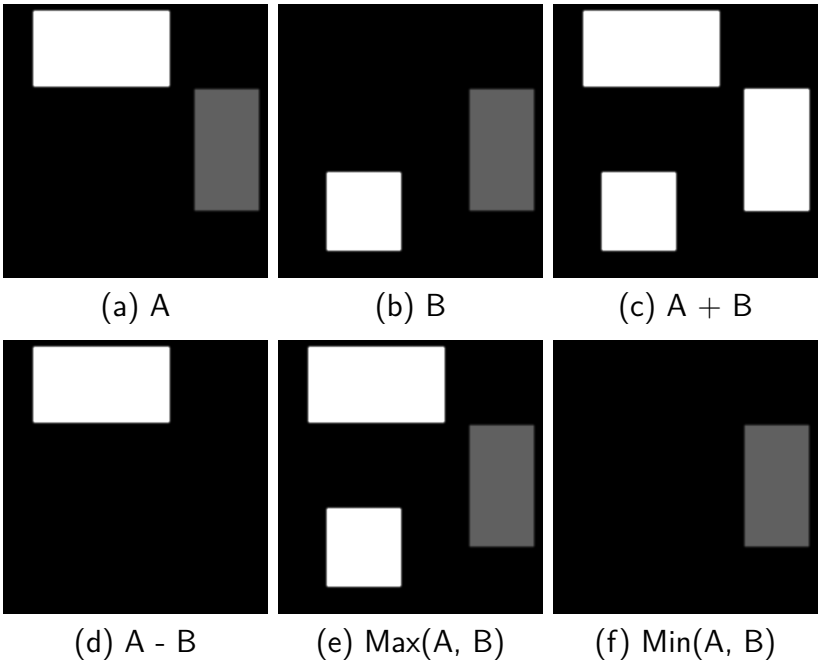

Figure 3.7: A few results from the reference tests on the synthetic energy bins A and B.

dataset. This dataset was created with Microsoft Paint and included three objects to represent three materials. There were two energy bins and each object had a value of 0, 0.5, or 1 in each of the energy bins. This way, the behaviour of each operator vould be verified from the output. A few of the results of this test are shown in Fig. 3.7.

The performance tests were evaluation tests. 4 frames per second (fps) was selected as the minimal acceptable interactive frame rate. Over the various tests performed on different datasets, the average frame rate was measured at 15fps for a 512x512x450x4 dataset. A more complete benchmark test was performed later on the final design which is presented in Chapter 5.

3.3.3 Final Design of the Intermixing Framework

The second design had a few important shortcomings. Firstly, the design criteria was not yet achieved as 4 operators were missing. Secondly, the expression construction system had an inflexible design. Expressions had to

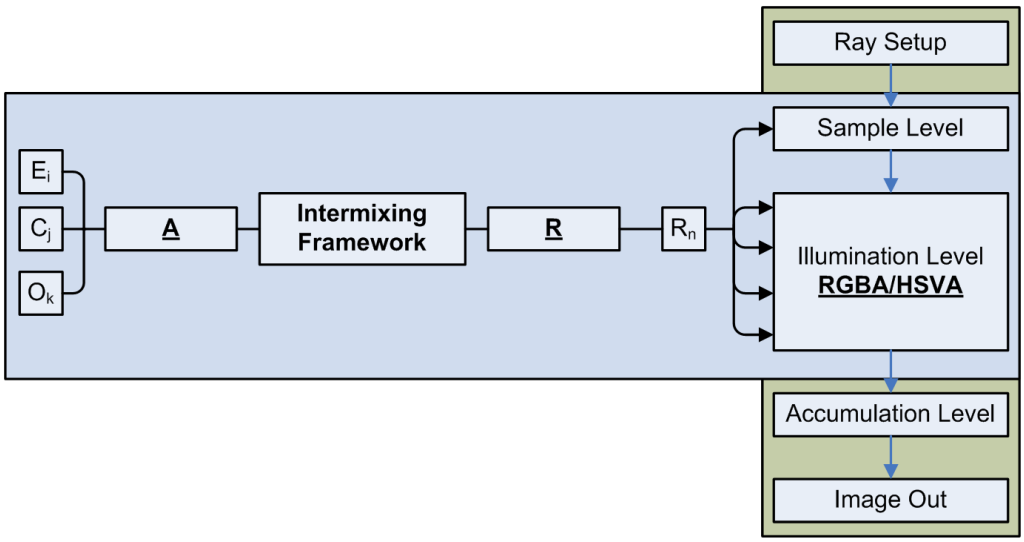

Figure 3.8: The system diagram of the final intermixing framework design.

be completely rewritten to accomodate any changes as the expression was a single string with no editable structure. The final design addressed both issues.

The system diagram of the final design is shown in Fig. 3.8. The user creates vectors of size four from energy bins, constants and operators. These vectors are processed by the algorithm construction system. The components of the resulting vector are extracted and applied to the rendering pipeline.

This system advances the previous system in a few ways. Firstly, this system completely implements all 15 operators based on the definition from equation 3.1. Secondly, the algorithm construction system is fully object-oriented to produce a robust and flexible system. Lastly, the mapping between the algorithm construction and the rendering pipeline is now explicitly defined.

Algorithm Construction

The new algorithm construction system is fully object oriented. The base object is the MARSOperand which provides methods to retrieve shader code. This object is then inherited by MARSOperator, MARSConstant, and MARSVolume objects which implement the retrieval code and store the relevant data.

The most important object is the MARSOperator. The MARSOperator object implements equation 3.1 by storing two MARSOperand vectors. The result is an operand tree similar to the volume tree defined by Woodring's system [1]. This is illustrated by Fig. 3.9. When the shader code is retrieved from the MARSOperator, the method recursively retrieves shader code from

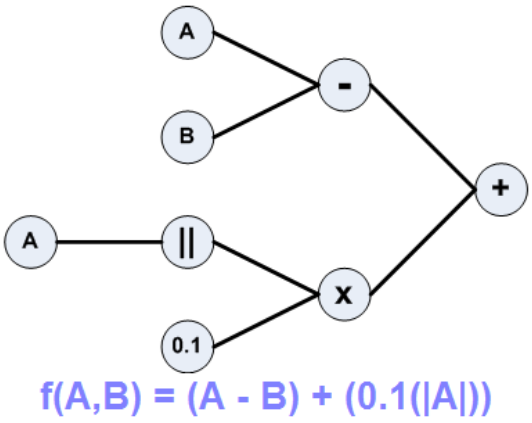

Figure 3.9: The operand tree is a structure formed by MARSOperand objects which include MARSConstants, MARSVolumes, and MARSOperators. Each MARSOperator contains two MARSOperands thus forming the tree branches. The leaves of the tree are MARSConstants and MARSVolumes.

rest of the operand tree. In this way, the expression string is created as needed.

This design has three important benefits. Firstly, the macro definitions are now redundant as each of the 15 operators are defined within the MARSOperator object. Secondly, an object oriented system allows each MARSOperand in the expression to be altered with minimal effort. Lastly, each MARSOperand object can be independently visualized allowing the user to visually navigate through each step of an expression. In addition, MARSOperand objects may also be shared between expressions improving the performance when rendering the result as the total number of operations decreases.

Mapping Data Structures to the Rendering Pipeline

Based on the design criteria, data will be intermixed at the sample and illumination levels. As can be seen in Fig. 3.8, this is defined by five scalar

channels to which data can be mapped. The mapping process happens in an object called MARSPipeline. When MARSShader generates the custom shader, MARSPipeline is accessed to retrieve shader code from each of the five data channels. The data channels are MARSOperands to allow MARSOperators, MARSVolumes, and MARSConstants to be valid data structures in the rendering pipeline.

The benefit of the MARSPipeline object is two-fold. Firstly, multiple MARSPipeline objects allow the user to store view configurations and easily switch between them. Secondly, the MARSPipeline object knows which data structure is applied at the sample level and at the illumination level. Therefore, a preprocessing step can ensure that each calculation or texture fetch only happens once at the optimum position in the shader thereby maximizing performance.

An issue exists between the MARSPipeline and the algorithm construction objects. MARSVolumes and MARSOperators both define vectors of size four. However, MARSPipeline only deals with scalar data channels. To simplify the mapping process, all MARSOperand data types are accessed component-wise. This prevents any clashes between the data structures and the rendering pipeline.

Transfer Functions as Operators

Transfer functions are usually considered to be look up tables for illumination parameters. However, mathematically the transfer function is a map from R to R^4. As the intermixing framework operates with vectors of size 4, the transfer function can then function as part of the intermixing framework

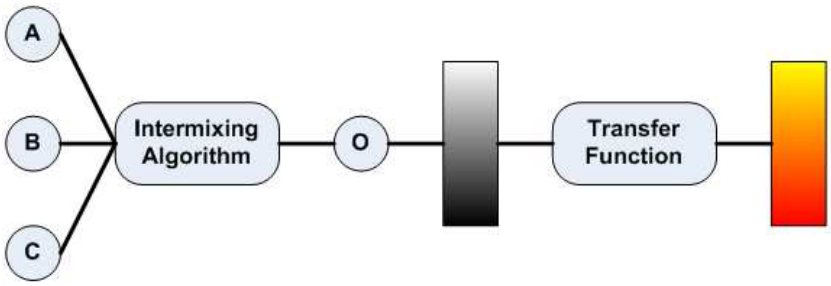

Figure 3.10: The transfer function can be a useful addition to the intermixing framework to quickly apply colour maps to results.

making it the 16th operator.

The motivation for this step is to simplify common visualization tasks. When a scalar dataset is visualized it is common practice to apply simple colour maps such as gradients. It is simpler to define a gradient transfer function then to create a mathematical gradient with the other 15 operators. This concept is illustrated in Fig. 3.10.

3.3.4 Summary of the Intermixing Framework

The final intermixing framework implements a flexible and generic design. The framework is built from two object-oriented systems including the algorithm construction and the shader construction systems. The complete structure is illustrated in Fig. 3.11.

The algorithm construction system is based on the operator definition in equation 3.1. The system uses inheritance and recursion between the MARSOperator and MARSOperand objects to create the shader code snippets as needed.

The shader construction system is based on dynamic shader generation. A large pool of code snippets defines the structure for raycasting, iso surfacing,

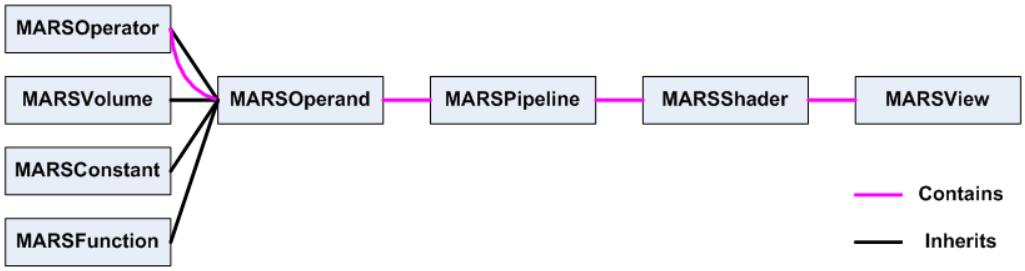

Figure 3.11: A basic diagram outlining all objects which make up the intermixing framework and the relationships between them.

maximum intensity projection, and 2D slicing algorithms. MARSShader then completes the code by pasting the generated code snippets from the algorithm construction system.

The link betwen the two systems is formed by the MARSPipeline object. This object explicity defines what data structures is passed to the sample and the illumination levels during rendering. In addition, the MARSPipeline object also optimizes shader code by preventing redundancy.

An addition to the intermixing framework is the definition of the transfer function as a MARSOperand. This allows basic transfer functions such as colour gradients to assist visualization of intermixing algorithms.

The final design of the intermixing framework was extensively tested with four spectral CT case studies and a formal benchmarking test. The results produced are revealed in Chapter 5.

Chapter IV

Visualization Software: MARSCTExplorer

This chapter describes the design and development of a visualization application called MARSCTExplorer. Firstly, design criteria is drawn up based on the aims of this research. This is followed by a detailed description of MARSCTExplorer covering the structure, tools implemented, and the graphical user interface.

4.1 Requirements for MARSCTExplorer

The final component of the intermixing framework from Chapter 3 is the graphical user interface. MARSCTExplorer is an application designed to implement, complete and evaluate the intermixing framework. In doing so, MARSCTExplorer aims to be able to answer two basic questions.

- Can the intermixing framework extract the desired material information from a given spectral CT study?

- How do the results of the intermixing framework compare to the traditional transfer function methodology?

The first question evaluates the value of the ability of the intermixing framework to visualize spectral CT data. The second question is an evaluation of the both the intermixing framework and spectral CT imaging to demonstrate that spectral CT imaging advances CT technology.

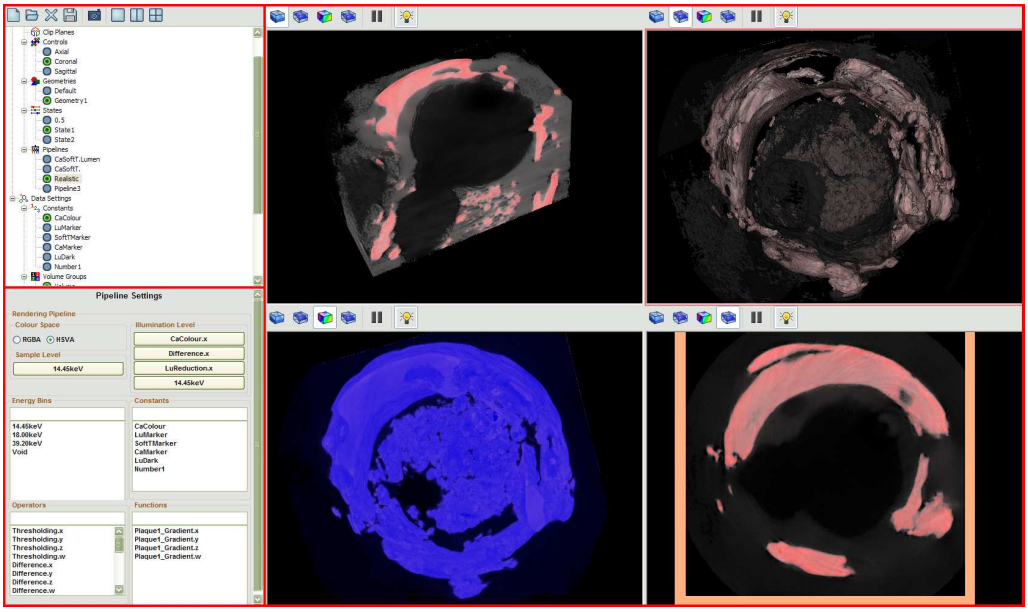

Figure 4.1: A screenshot detailing the layout of MARSCTExplorer. The three primary regions are highlighted with red boxes. Top left; the study tree. Bottom left; the property notebook. Right; the viewing area.

From these questions, some basic criteria can be drawn for the design of MARSCTExplorer. Firstly, MARSCTExplorer must implement the intermixing framework from Chapter 3. Secondly, MARSCTExplorer must implement traditional visualization techniques and tools. Thirdly, MARSCTExplorer must provide sufficient tools to interact with all results. Lastly, the graphical user interface (GUI) should be as simple and intuitive as possible.

4.2 Basic Interface of MARSCTExplorer

Fig. 4.1 shows the basic layout of MARSCTExplorer. The graphical user interface is divided into three regions including the viewing area, the study tree in the top left, and the property page in the bottom left.

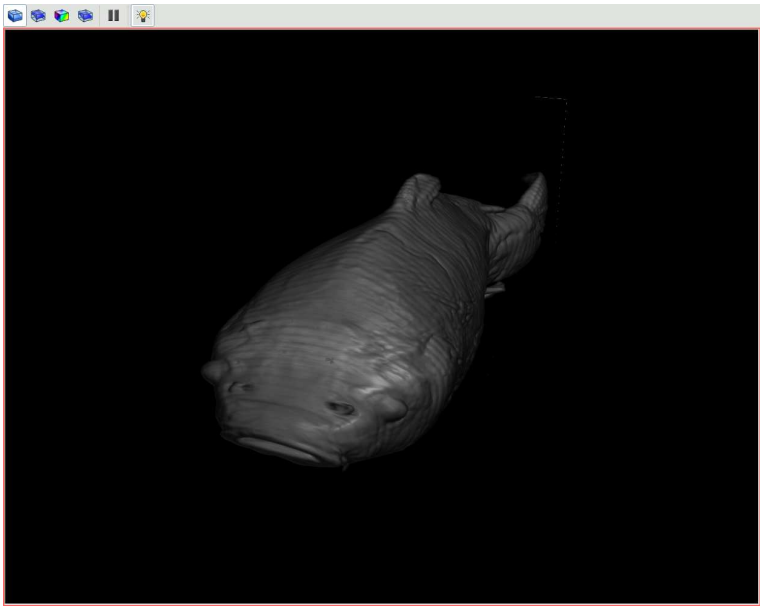

Figure 4.2: The views consist of a toolbar, a rendering canvas, and a border to highlight the active view.

4.2.1 Viewing Area of MARSCTExplorer

The viewing area has three configurations including a single view, two views split horizontally, or four views. In this way spatial visualization is possible. In addition, if the system is configured with two monitors as a single virtual screen then MARSCTExplorer adds another viewing area giving a maximum of eight views.

Each view is defined by a canvas and a toolbar as shown in Fig. 4.2. The toolbar switches between the different rendering algorithms. In addition, there is a pause button to toggle motion on/off while the end button toggles shading on/off.

The main canvas is the rendering viewport with a perspective projection system. Rendering is performed only when necessary such as motion events or parameter changes. The view also supports active stereo using NVIDIA's

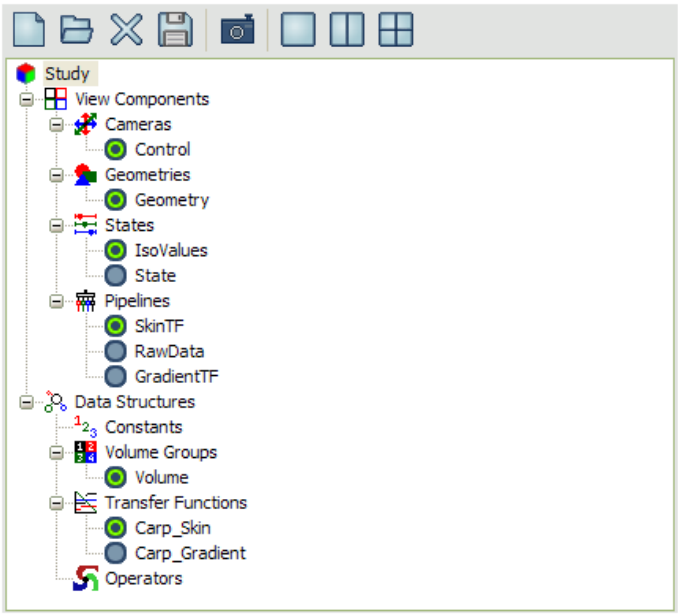

Figure 4.3: This tree structure defines all controls, geometries, states, pipelines, transfer functions, volumes, constants, and operators. The user selects the active view components and edits the active data components.

3D Vision technology. Lastly, the canvas has a border to reveal which view is active.

4.2.2 Study Tree for MARSCTExplorer

MARSCTExplorer is based on a study concept. A study is defined by data structures and view components as illustrated by the study tree in Fig. 4.3. The view components include controls, geometries, states, and pipelines. The data structures include volumes, constants, transfer functions, and operators.

The study tree achieves three main goals. Firstly, the study tree provides an overview of the study contents grouping each component by its type and association. Secondly, the study tree reveals which components are active via the small icons next to each tree entry (green means active). Lastly,

the study tree switches the property page each time a new component type is clicked. This allows the interface to hide all unnecessary tools, thereby keeping the visuals clear and simple.

Another important feature of the study tree is the ability to rename components. For example, a constant used to threshold a calcium boundary could be called "*CaThreshold*". This allows the user to use logical naming systems to improve the clarity of the study.

4.2.3 Property Pages for MARSCTExplorer Components

The property pages define all settings and tools. In short, the functionality of MARSCTExplorer is almost completely defined inside the property pages. A unique page exists for each study component type and the role of the property page is to show the correct tools when required, and hide all tools which are not required. This way a large amount of functionality is accessible with little to no visual clutter.

A typical property page is shown in Fig. 4.4. This property page is for the control component which manipulates the camera as a trackball with the centre of rotation as the centre of the volume. The tools shown here control the home position and the current offset and rotation. These are also updated when the view is panned, scaled, or rotated using the mouse.

4.3 Standard Toolset of MARSCTExplorer

The standard toolset in MARSCTExplorer is implemented both as explicit tools and via interaction between the component objects and the views.

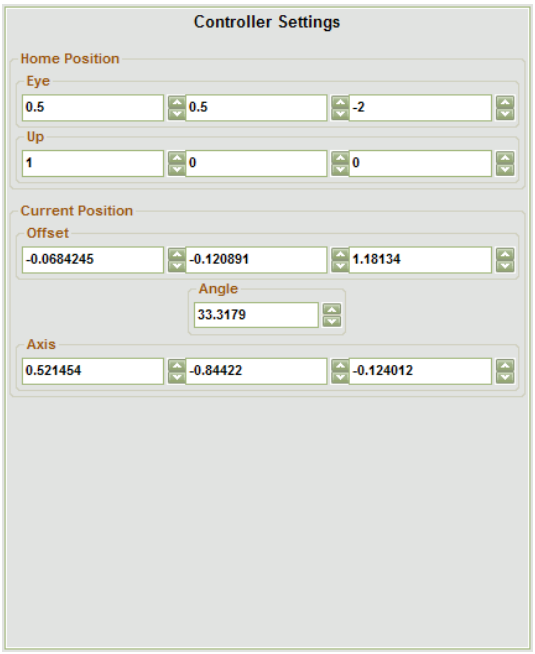

Figure 4.4: Each property page defines the set of tools associated with the study component selected. This page shows the tools for the control component which controls the camera.

4.3.1 Explicit Toolset for MARSCTExplorer

The explicit tools define the functionality within each view component. The property pages for each view component provide widgets to access the relevant tools.

Control Components

As seen in Fig. 4.4 earlier, the control component defines the home position and the current offset and rotation from that position. In addition the control also provides the pan, scale, and rotate functionality inside the views with the mouse.

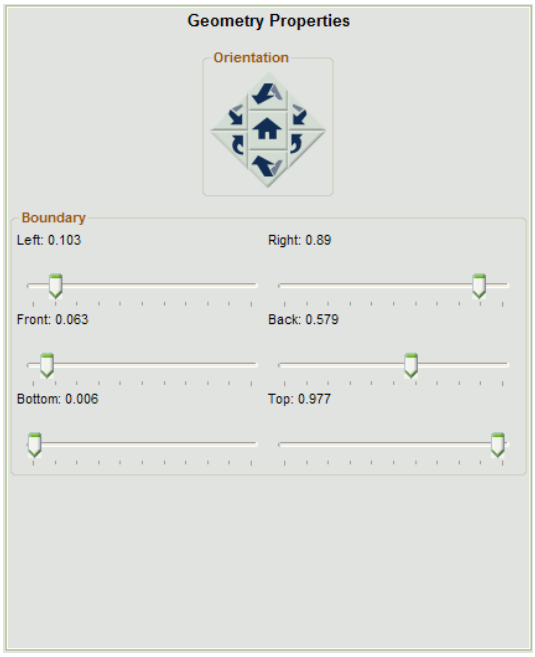

Figure 4.5: The geometry page can rotate the geometry as well as adjust the bounding box within a unit cube.

Geometry Components

The geometry component defines the bounding box of the volume. There are two tools provided for the geometry as seen in Fig. 4.5. The first is a rotation of the geometry itself (as opposed to camera rotation using control components). This allows two views to be orthogonal to each other even though the same control is used.

The second tool controls the geometry boundaries within a unit cube. The boundaries are used to define the start and end positions of the ray during raycasting. Access to textures can only occur within the range 0.0 to 1.0 thereby restricting the geometry to the unit cube. Changing the geometry boundaries allows clipping of volume on the axis as shown in Fig. 4.6.

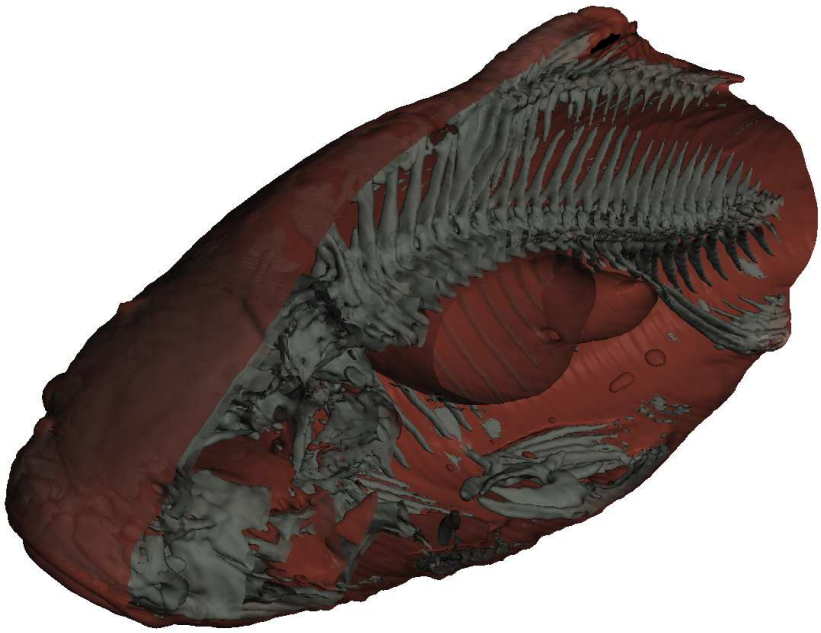

Figure 4.6: The bounding box is a good way to clip the volume along the axis.

State Components

The state component contains the volume rendering parameters as shown in Fig. 4.7. Internally the state is simply a uniform array latched to a series of slider bars. Each slider bar provides control over a single parameter.

The first parameter is the maximum number of iterations over maximum possible ray which has length $\sqrt{3}$. A much better algorithm would divide the volume into segments and calculate the required samples for each segment based on the Nyquist frequency. Still the time spent to implement adaptive sampling outweighs the gain in performance which would be achieved.

The following two parameters are thresholding parameters defining both high and low thresholds. When the lower threshold is less than the higher threshold the active dynamic range is between them but when the lower

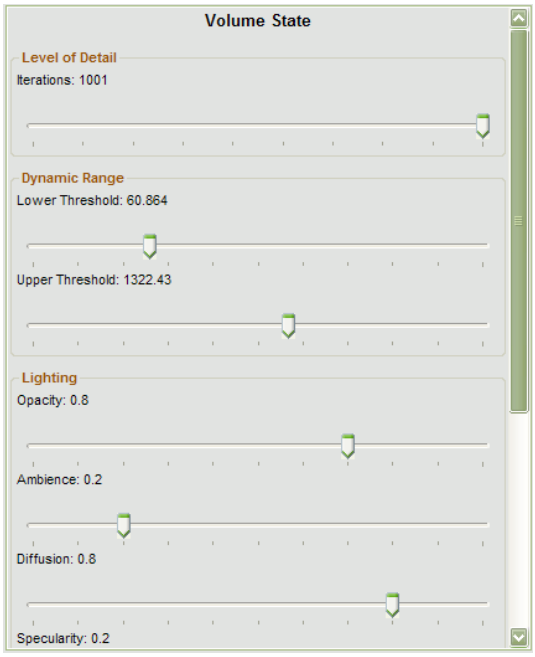

Figure 4.7: The state controls the volume parameters. This includes the number of iterations, thresholding, global opacity, lighting parameters, and screen parameters.

threshold is greater than the higher threshold the active dynamic range is outside them. Alternatively, the threshold parameters become iso-values when iso-surfacing and become the windowing bounds for the 2D slice viewer.

The forth parameter is the global opacity which is a simple way of increasing visibility through the volume.

The following three parameters are the ambience, diffusion, and specularity parameters for the Blinn-Phong lighting model [10]. There is only one light which is fused to the camera. While external lights are more realistic, a camera light simplifies rendering while providing a sufficiently clear image for visualization.

The last three parameters are screen parameters including the brightness, contrast, and gamma. While not necessary, these tools can quickly enhance

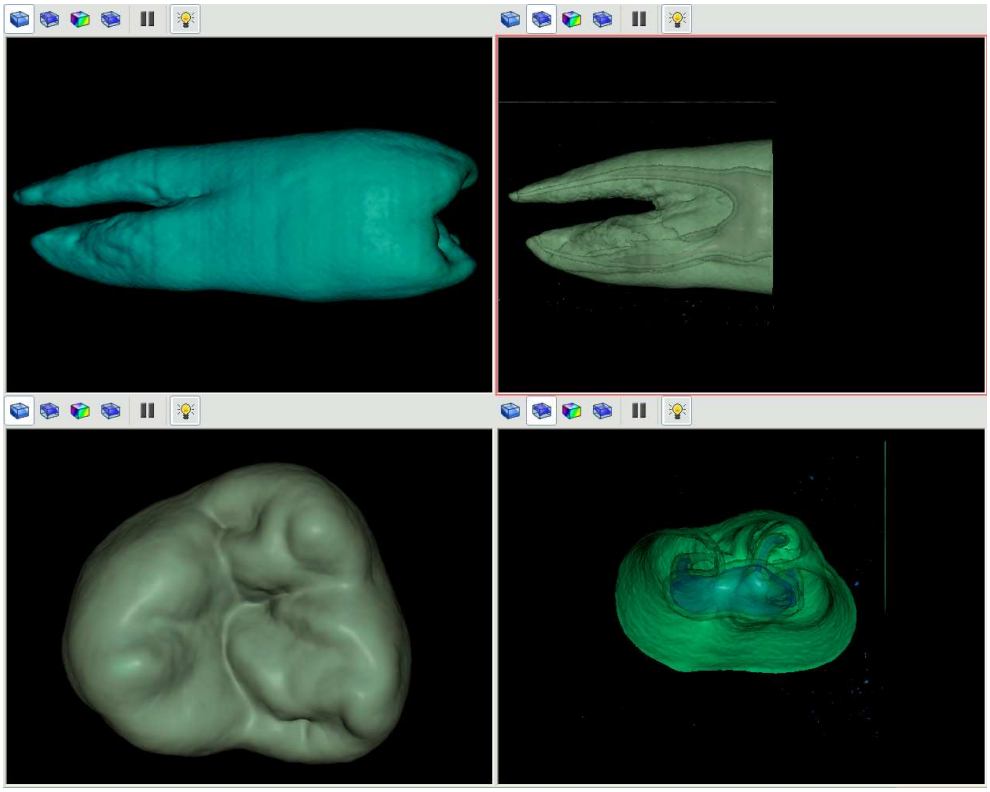

Figure 4.8: As each view component attaches to the view scenegraphs, the components may be shared and hence the effects shared.

visibility of dark regions and dull colour shades.

4.3.2 Implicit Toolset for MARSCTExplorer

Each study component is a separate object which can be added or removed from the views. The view components attach directly to the view scenegraphs while the data structures attach to the pipeline components. This provides some additional functionality as demonstrated in Fig. 4.8.

Fig. 4.8 shows the outcome when two controls, two geometries, two states, and two pipelines are shared by two out of the four views. Sharing controls synchronizes the orientation as revealed by the top and the bottom views.

Sharing geometries fuses the boundary conditions as shown by the views of each side. Sharing states joins the volume rendering parameters and sharing pipelines means that the data content of the views are identical as demonstrated by the views along each diagonal.

4.4 Intermixing Framework Interface

The intermixing framework is implemented via the data structures as well as the pipeline components. The data structures define all available data while the pipelines then attach the data to the view scenegraphs.

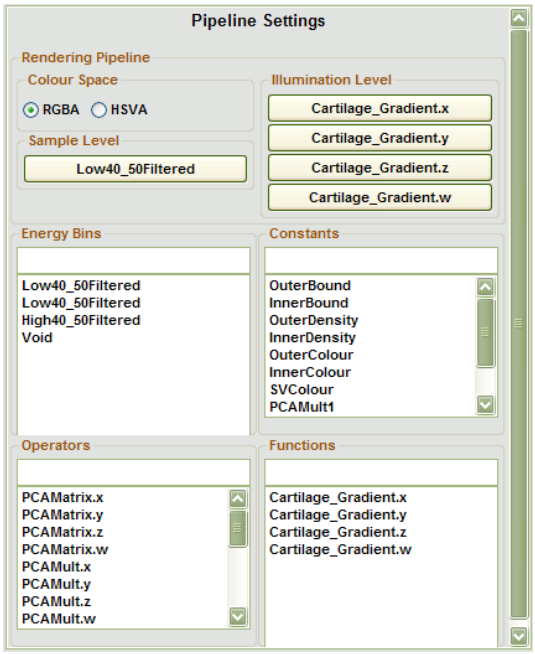

Figure 4.9: The pipeline page selects the illumination model and fills the five data channels in the render pipeline.

4.4.1 Interface Between Views and Data Sources

The pipeline components map the data structures to the rendering pipeline as shown in Fig. 4.9. Recall that the rendering pipeline has 5 data channels including the sample level channel and the four illumination level channels. The user selects the relevant data structures from the lists of energy bins, constants, transfer functions, and operators and applies the data structure to the relevant channel via the channel buttons. In addition the user can also toggle the illumination model between the RGBA and HSVA models.

4.4.2 Data Structures of a MARS study

The data structures consist of constants, energy bins, transfer functions, and operators. Each of these has parameters which can be altered and/or analysis

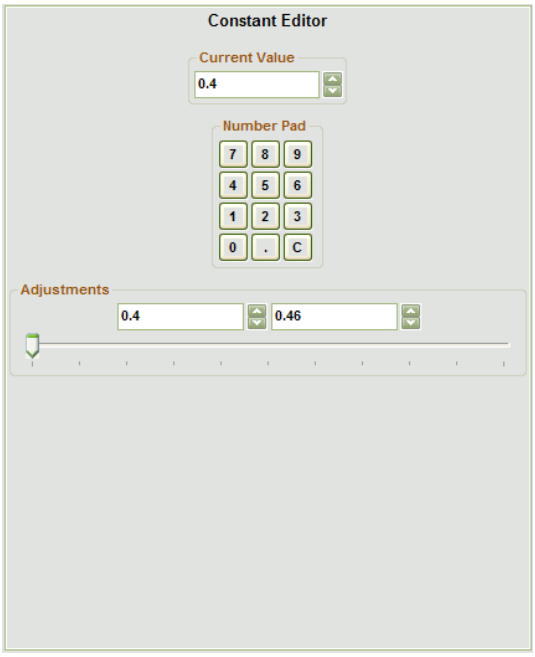

Figure 4.10: The constant page contains a few methods to control the current value as well as the minimum and maximum boundaries.

tools.

Constant Components

The first data structure is the constant. The constant contains a single value together with minimum and maximum boundaries. The constant property page is shown in Fig. 4.10. The top widget shows the current value and allows the new values to be typed. The next widget then allows the value to be punched in via a calculator interface while the bottom widgets allow the value to be updated via a slider. In addition, the bottom widget also contains the minimum and maximum values.

The slider bar is the most important widget in the constant property page. When a constant is used in an intermixing algorithm, it means that the slider is mapped to the equation providing interactive control over the result. A particularly useful feature is that the slider ranges from the minimum to the maximum value thereby also providing control over the slider's precision.

Energy Bin Components

The second data structure is the energy bin. This is the raw data and does not currently have any parameters to edit. Instead, the property page contains data analysis and filtering tools as shown in Fig. 4.11.

The data analysis tool is a PCA calculation utility. The user selects how many random samples to take for the calculation then performs PCA to find the transformation matrix. The resulting matrix can then be automatically applied to the data as an operator by clicking the "Make Operator" button.

The PCA calculation is done with a CUBLAS implemented PCA algo-

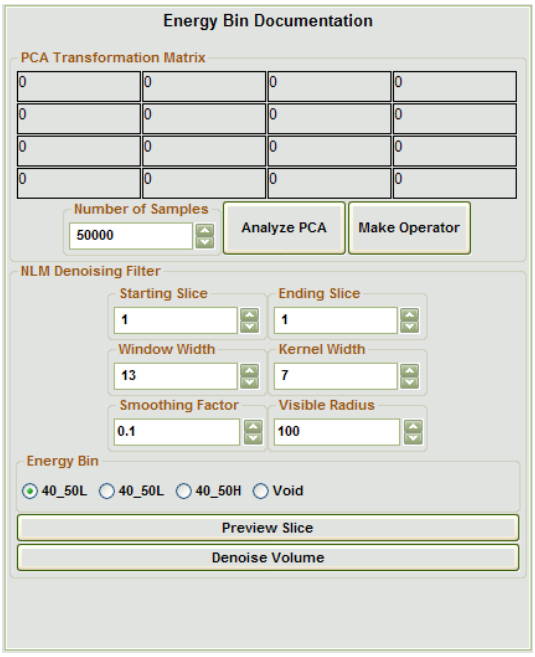

Figure 4.11: The energy bin property page contains a PCA calculation tool as well as a non-local means denoising filter. Both are CUDA accelerated.

rithm [27]. This means that for most cases the calculation takes less than 10 seconds depending on the number of samples chosen. As the transformation matrix is applied to the data in real-time, this means that PCA is now a rapid and flexible data analysis tool. The only limitation is that the matrix is limited to size 4 as imposed by GLSL.

The filtering tool is a CUDA implementation of the non-local means filter. The user first selects a target slice with the starting slice widget and sets initial parameters. The preview button then denoises and shows the selected slice as in Fig. 4.12.

The most important image in the preview is the difference image which shows what data was lost (the image is scaled to the full dynamic range to provide maximum visibility). If the parameters are well selected then the

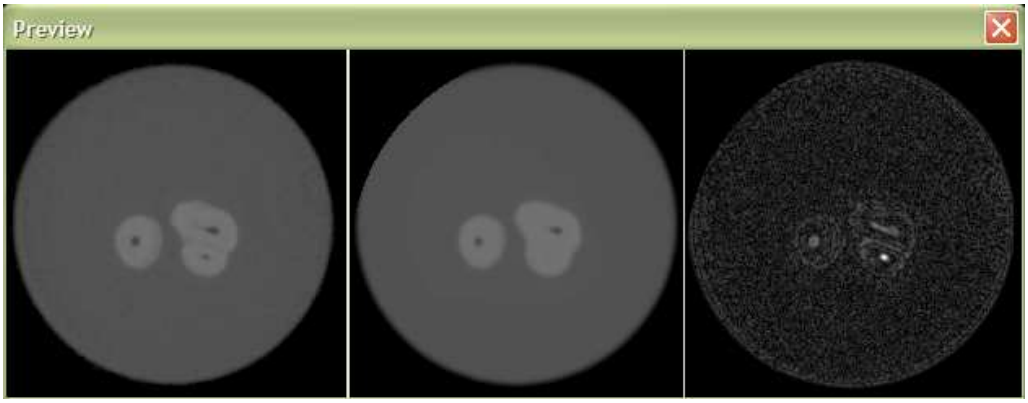

Figure 4.12: The non-local means preview shows the original, filtered, and difference images for a single selected slice. This allows quick alterations of the algorithms parameters.

difference image should be pure white noise. Otherwise, if structures can be seen in the difference image, then the parameters need to be altered. Once the difference image is at an acceptable level, the filter then procedes on all slices between the starting and ending values.

The non-local means filter usually processes an image within 8 seconds. Therefore, a whole volume containing 512 slices can easily take over an hour. The resulting filtered and difference images are written to separate directories within the study directory structure.

The study directory structure is a base directory for the study and separate internal directories for each energy bin. The study itself is defined by a .sty file and transfer functions are also stored as separate .tfn files. Both the .sty and .tfn file types are simple text files adopting a basic scripting language. For more information see Appendix C.

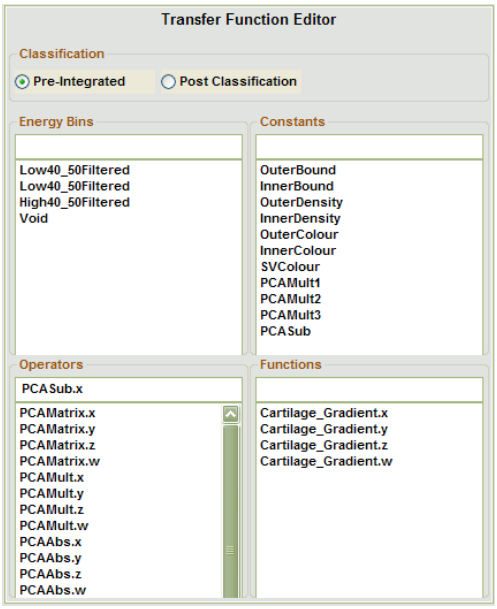

Figure 4.13: The transfer function page allows the transfer function to operate on energy bins, constants, operators, or even other transfer functions.

Transfer Function Components

As described in Chapter 3, the transfer function is an active component in the intermixing framework. The resulting property page is shown in Fig. 4.13.

The immediate observation is that the transfer function editor is missing. It was judged that the time taken to implement the transfer function editor was not acceptable within the scope of this research. Therefore, the transfer functions are currently edited manually in the .tfn text file.

Even though the transfer function editor is missing, the other tool does provide some unique benefits. The rest of the transfer function property page allows the user to select the data structure which the transfer function operates on. This allows the transfer function to operate on energy bins, constants, operators, or even other transfer functions. Lastly, MARSCTExplorer

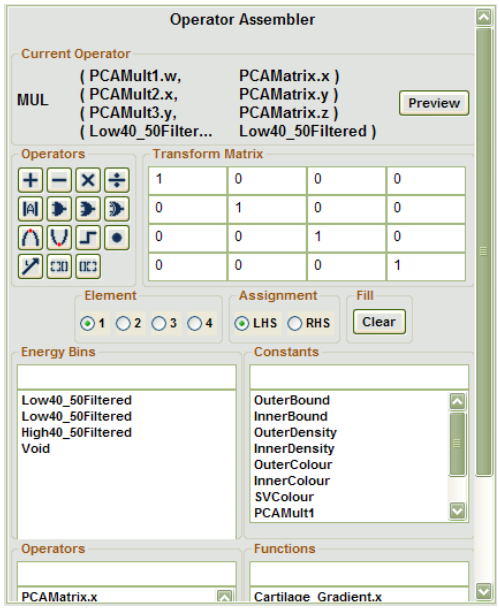

Figure 4.14: The operator page shows the current expression, provides a preview of the active expression, and allows all parts of the operator to be altered.

supports both post and pre-integrated classification.

Operator Components

The operator components are the core of the intermixing framework as each intermixing algorithm is defined by expressions which are constructed by operators. The resulting property page is shown in Fig. 4.14.

The top of the operator page shows the current expression. The user selects a component and side to edit. The targeted operand is then altered by selecting a data structure or by clearing the field. In addition, the preview button uses the active view to preview the active component of the operator.

The operation performed by the operator component is set by the 15 buttons arranged in a calculator-style interface. Next to these buttons is the transformation matrix which is only active if the operator is a transformation.

The true potential of the operators are brought out by careful design. For example, if there are multiple independent additions in an algorithm, then up to four additions can be grouped into one operator. If the operator defines a specific volumetric parameter such as colour, then a useful operand would be a constant to latch the constant's slider to the algorithm. Finally, careful naming of the operator improves understanding and also assists the mapping process to a pipeline.

The operator is the least user friendly component in MARSCTExplorer. On the other hand, the operator provides the most power and control over the spectral CT data. Most importantly, however, is that the design is such that an experienced user should be able to design algorithms so that novice users only need to tweak sliders in the constants to get the results they want without understanding the operators used.

4.5 Summary of MARSCTExplorer

The intent of MARSCTExplorer was to implement the intermixing framework together with sufficient tools to support a complete evaluation of the framework and its results. During the design of MARSCTExplorer, great care was taken to ensure that the resulting interface was as user friendly as possible. This was achieved by using a modular, object oriented design with an intuitive structure.

Unfortunately, due to time constraints, no user studies could be performed to evalute and improve the graphical user interface of MARSCTExplorer. Still, the structured interface is sufficient for a research-based application.

Chapter V

Evaluation of MARSCTExplorer

This chapter introduces four spectral CT case studies to evaluate the power and performance of MARSCTExplorer. Each study targeted specific materials and MARSCTExplorer then identified and exposed those materials. In addition, comparisons were made between the intermixing framework and transfer function based visualization techniques. Lastly, a simple benchmarking test validated that MARSCTExplorer operates with interactive framerates.

In all case studies the data was cropped to the minimal possible size to maximize performance and the NLM filter was applied to maximize quality.

5.1 Case Study I: Phantom for Colloidal Gold

The phantom was a study of gold nano-particles acquired by Raja Aamir. Five samples with varying concentrations of colloidal gold and Omnipaque were contained in a perspex frame. The sample contents are listed in Table 5.1. The goal of this study was to separate the samples. To achieve this, 24 energy bins were acquired with constant threshold intervals between 447 and 725 (which corresponds to 5.59keV and 46.2keV respectively). The useful data was contained in an 8-bit volume of $280 \times 280 \times 20$ voxels.

Fig. 5.1 shows the results achieved with MARSCTExplorer. Energy bins

Table 5.1: The various samples of colloidal gold and Omnipaque.

Sample Number	Sample Content
1	Colloidal gold
2	Omnipaque (3ml/100ml)
3	50% Omnipaque, 50% gold
4	80% Omnipaque, 20% gold
5	20% Omnipaque, 80% gold

1, 8, 16, and 24 were selected by maximum spread and PCA was performed. One of the resulting components provided good contrast between the samples. The constrast between the samples was scaled to the full dynamic range and applied as the hue to get the colour scheme shown.

The result shows unique colours for all five samples although samples 3

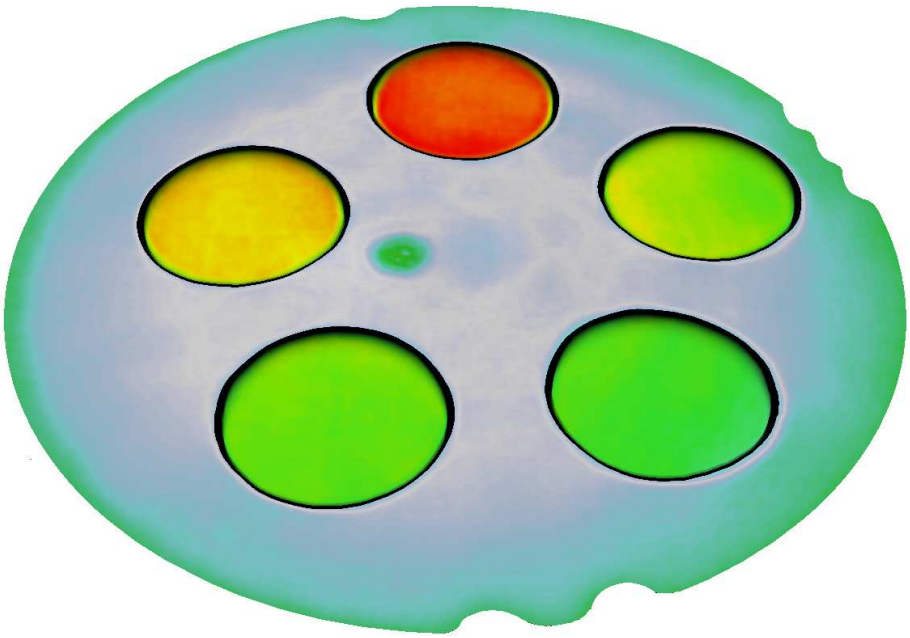

Figure 5.1: One component from PCA on four of the energy bins contained good contrast between the samples. Scaling the contrast further and applying the result to the HSVA illumination model produces the colour scheme shown. Sample numbers count clockwise from the top.

and 4 have extremely similar colours. Comparing the top portion of sample 4 and the bottom portion of sample 3 reveals the difference as sample 3 fades to turquiose while sample 4 has a yellowish tinge.

Another feature of Fig. 5.1 is the unique style achieved. The geometry was defined by boolean masks created with the step function. This means that shading is only valid for the boundaries as gradient has magnitude zero elsewhere. By setting the diffusion, ambience, and specularity values to zero, the shading is reduced to solid black creating the outlines shown.

To summarize, the goal of separating the samples was successful. In addition, the results reveal how the intermixing framework allows unique rendering styles by manipulating shading effects.

5.2 Case Study II: Cartilage Density

The cartilage study, acquired by Annamie Seigert, aims to measure Glycosaminoglycan (GAG) content in cartilage. The initial task was to gain an overview of the density distribution of the cartilage based on the injection of Hexabrix. Hexabrix is a contrast agent which binds to the GAG with an inversely proportional relationship.

The study targets four concentrations of Hexabrix over two datasets with two energy bins each. Each dataset contains four pieces of bone with two pieces for a single concentration of Hexabrix. The different Hexabrix concentration samples are separated by a microscope slide. One dataset contains 20% and 30% concentrations of Hexabrix while the second dataset contains 40% and 50% concentrations of Hexabrix. The useful data was contained in an 8-bit volume of $664 \times 664 \times 254$ voxels.

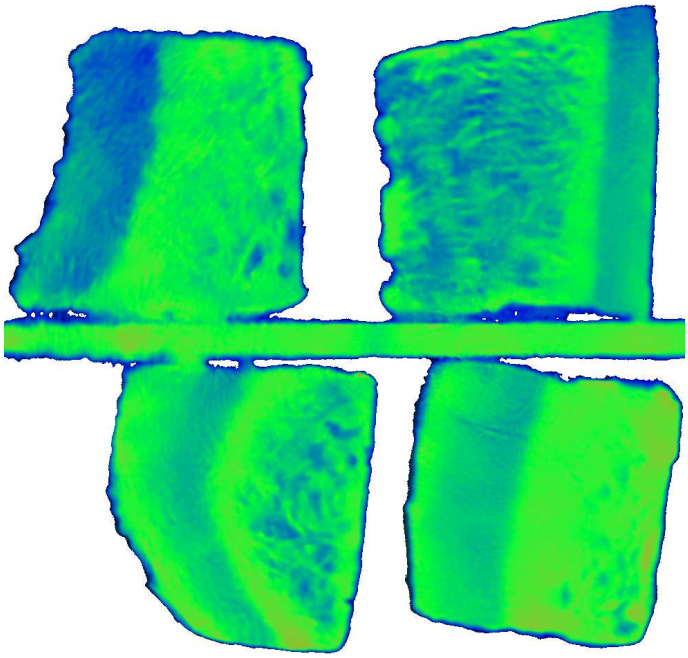

Figure 5.2: A gradient applied to a thin section of the first PCA component provides an initial overview of the density distribution of the cartilage. The two top pieces contain 20% Hexabrix while the bottom pieces contain 30% Hexabrix.

The goal for visualization was to compare the effects of the Hexabrix concentrations on the resulting datasets. Fig. 5.2 shows the results for the 20% to 30% dataset while Fig. 5.3 shows the results for the 40% to 50% dataset. In both cases, PCA was performed on the two energy bins per study to maximise contrast over the bone pieces. The results were then applied to a transfer function defining a gradient from blue to red through green.

The two images show that increases in the Hexabrix concentration visibly increase the contrast in the cartilage. This is shown in both images by the increased presence of red in the cartilage for the higher concentration bone pieces.

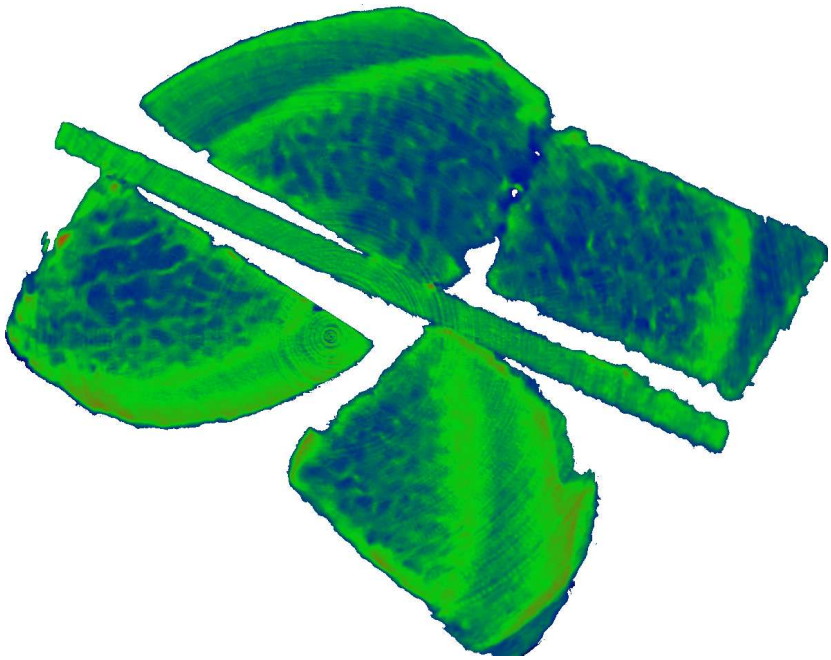

Figure 5.3: A gradient applied to a thin section of the first PCA component provides an initial overview of the density distribution of the cartilage. The two top pieces contain 40% Hexabrix while the bottom pieces contain 50% Hexabrix.

Another outcome of this result is the use of the transfer function as an operator in the intermixing algorithm. Applying the gradient transfer function to the PCA transformed data in real-time allowed a quick and simple colour map to enhance the visualization. This demonstrates that transfer functions are a valuable component to an intermixing framework. In fact, a list of simple transfer functions could function as presets for specific tasks e.g. density maps.

To summarize, the intermixing framework successfully provided an initial overview of the effects of Hexabrix concentrations on the contrast available in cartilage imaging. In addition, the results demonstrate how transfer functions can become important assets to intermixing algorithms.

5.3 Case Study III: Mouse 12

Mouse 12, acquired by Dr. Nanette Schleich, is a study in distinguishing the contrast agents iodine and barium. The mouse sample was prepared with an injection of iodine into the right heart chamber and the insertion of barium into the esophagus. In addition, calcium was already present in the bones. The sample was scanned over the thoracic area and the goal is to clearly distinguish all three primary materials.

The mouse study contains four energy bins at 15keV, 23keV, 30keV, and 35keV with the useful data contained in an 8-bit volume of $1024 \times 1024 \times 450$ voxels. Due to size limitations this was then scaled to $512 \times 512 \times 450$ voxels.

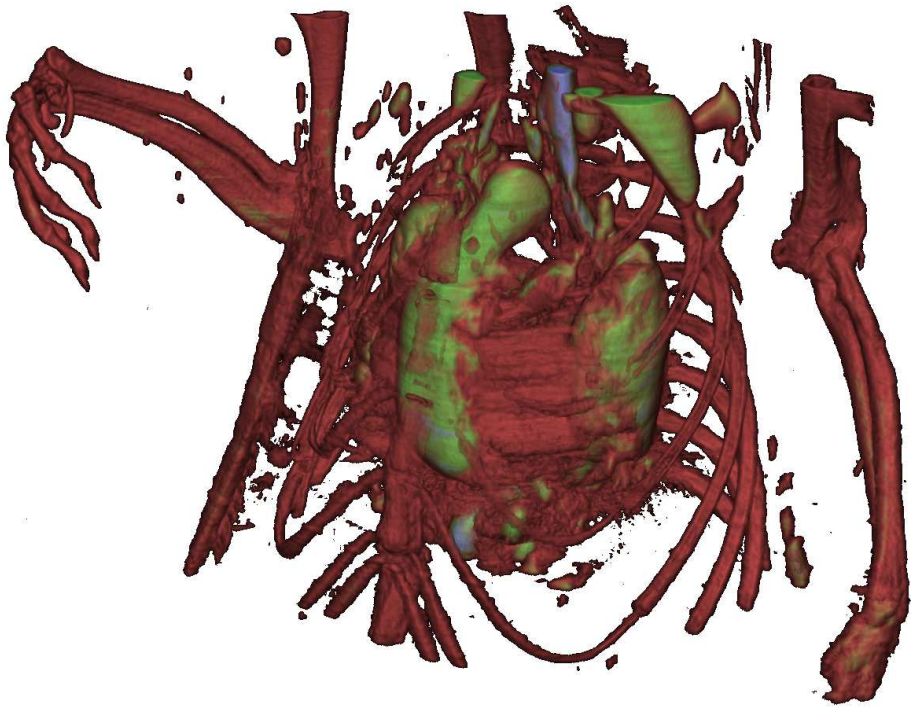

Figure 5.4: A transfer function applied to the broad spectrum to represent non-spectral CT imaging. Note the colour bleeding in the bones and the heart.

The k-edges of iodine and barium are at 33.2keV and 37.4keV respectively. This means that the difference between energy bins 30keV and 35keV should produce the greatest contrast between iodine and barium.

The first result in Fig. 5.4 is a control based on non-spectral CT imaging. The broad spectrum (lowest energy bin) is applied to a transfer function. Although the hand-made transfer function was not optimal, the results still show important features of the data. A high contrast colour scheme was adopted for clarity. In this colour scheme, red represents calcium, green represents iodine, and blue represents barium.

The first observation of Fig. 5.4 is that all three materials are visible. However, there is excessive colour bleeding as green (iodine) is visible in the

Figure 5.5: 15keV, 30keV and 35keV energy bins applied to the RGB channels directly. Note that the three primary organs are shown cleanly with no bleeding artifacts.

bones (red), blue (barium) is visible in the heart (green), and the esophagus contains layers of blue, green and red. The conclusion is that the non-spectral CT result roughly shows where each material can be found but doesn't cleanly distinguish them.

The second result in Fig. 5.5 uses the differences between the energy bins. The 15keV, 30keV and 35keV energy bins are directly applied to the red, green, and blue channels. Therefore, each colour directly corresponds to the differences between the energy bins.

In Fig. 5.5, all three materials now have a unique colour with no colour bleeding. All primary organs are a solid shade corresponding to the respective material. This is a significant improvement over the previous result in Fig. 5.4.

The biggest issue with algorithm used for Fig. 5.5 is the lack of flexibility. The colour choice is limited and are very similar in some cases. Initial observation could misinterpret the left lung as having the same colour as bone.

The final result in Fig. 5.6 solves the flexibility issue from Fig. 5.5. Here PCA was applied to all four energy bins to separate out some of the materials. Components one and three roughly correspond to the calcium and iodine respectively. Scaling the two components and applying the result to the RGB channels together with the broad spectrum results in the image shown. The scaling constants control the colours used in the final result.

Fig. 5.6 shows all three materials without colour bleeding and in high contrast colours. While, PCA has increased the visible noise in the result, this is negligible compared to the clarity of the separation of the three materials.

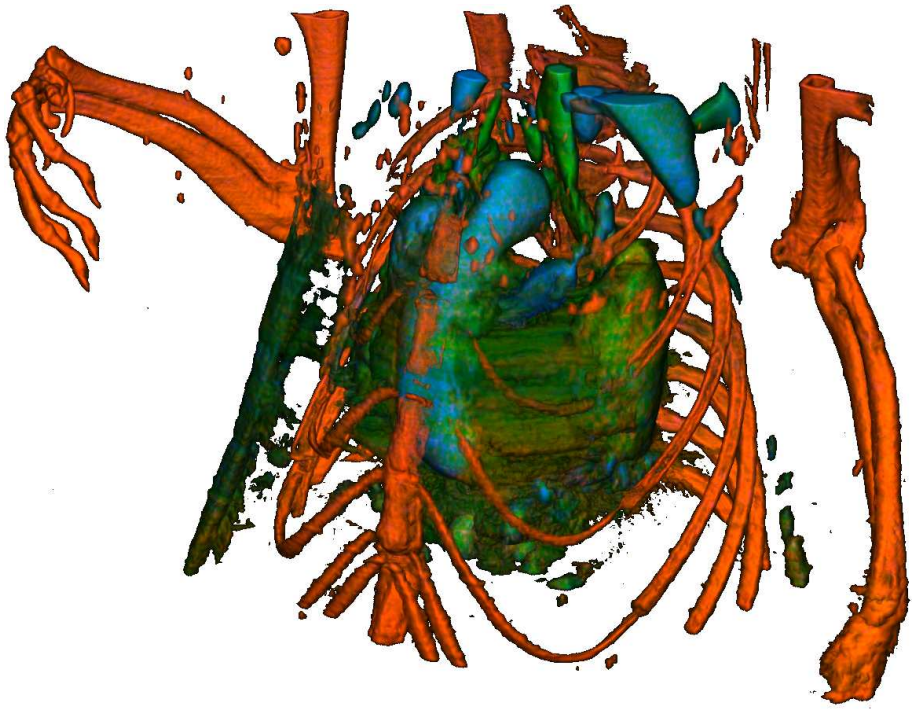

Figure 5.6: Broad spectrum and PCA components 1 and 3 applied to the RGB channels. The system can once again adopt a high contrast colour scheme while maintaining a clean separation of the three key materials.

To summarize, our intermixing framework was able to cleanly distinguish iodine, barium, and calcium using two different techniques. The second technique used PCA and scaling factors to provide better control over the outcome and reveal the result with a clear, high contrast colour scheme. In addition, non-spectral CT visualization was performed on the broad spectrum, and the results produced were inferior with colour bleeding between all three key materials.

5.4 Case Study IV: Human Excised Atheroma Plaque

Atherosclerosis is one of the most common causes of death for white males in the world. Unfortunately, not enough is known about the disease origins and

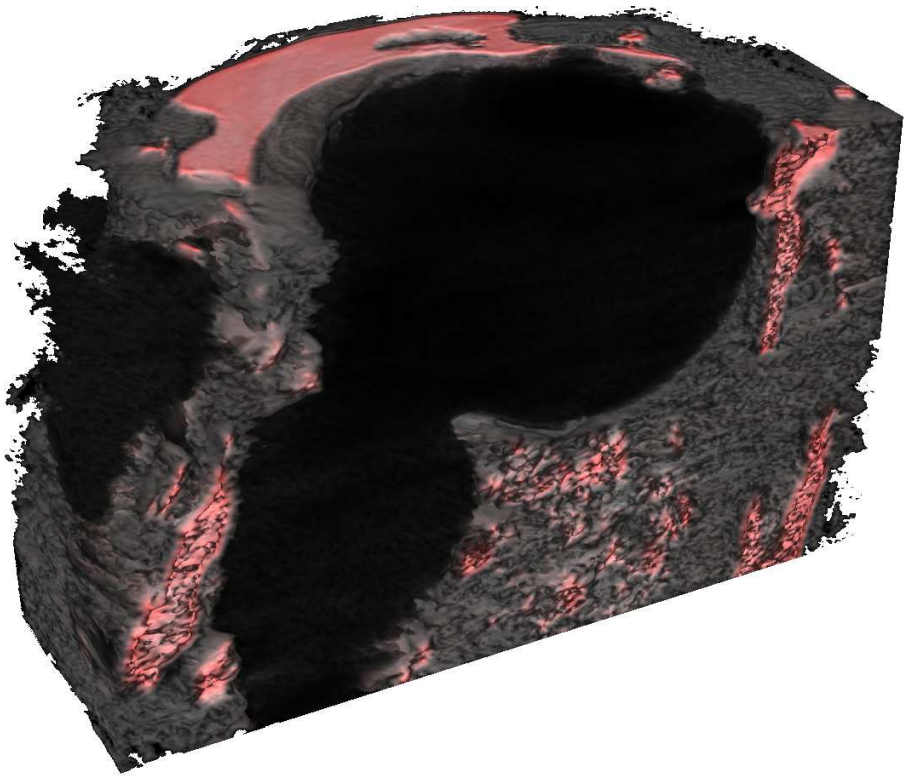

Figure 5.7: Calcium defined by applying the difference between the highest and lowest energy bins to the HSVA illumination model.

progression to significantly reduce this number. Therefore, athersosclerosis is an open field of research. The final case study, acquired by Rafidah Zainon, is a spectral CT dataset of a human excised atheromatous plaque.

The human excised atheromatous plaque study contains 8 energy bins between 14.45keV and 39.20keV with the useful data contained in a 12-bit volume of $361 \times 361 \times 181$ voxels. The initial aim of this study was to identify calcium with spectral means. The hypothesis was that the difference between the lowest and highest energy bins would reduce all materials significantly more then calcium. Therefore, the result can be assumed to be a good representation of the calcium distribution.

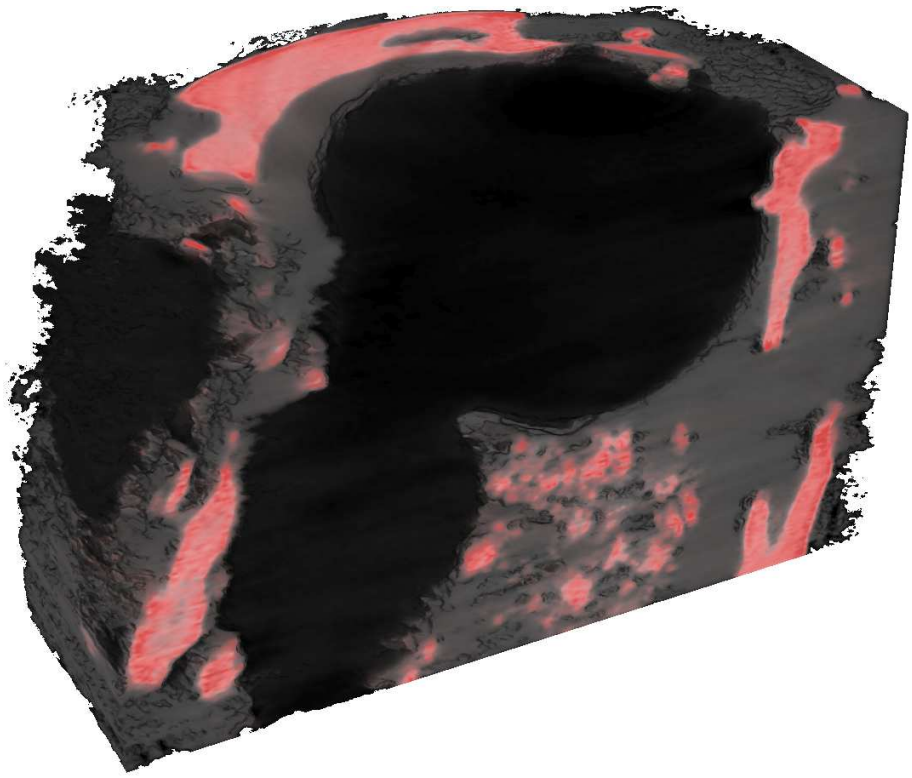

Figure 5.8: By applying a boolean mask to the sample level, the lighting was limited to the boundaries to provides greater clarity.

The first result in Fig. 5.7 applies a constant to the hue, the difference between the lowest and highest energy bins to the saturation. This means that the calcium should be revealed as shades of colour while the rest of the volume is grey scale. In addition the lumen was darkened by subtracting a boolean mask from the broad spectrum. This mask was defined by the step function and scaled by a constant to give control over the lumen darkness.

The resulting image in Fig. 5.7 shows the lumen (black volume) and calcium (red shades) quite clearly. In addition, the soft-tissue and epoxy are loosely separated with thresholding. The clarity of the calcium in this image means that the goal of identifying calicum was successful (still needs

histological verification) and that the concept of taking the difference between the highest and lowest energy bins is an effective strategy.

The second result in Fig. 5.8 provides an example of non-photorealistic lighting effects. Here, boolean masks (created with the step function) define the valid regions of the volume i.e. the lumen, soft-tissue and calcium. Applying these regions to the sample level means that shading only occurs on the boundaries. The result is a cleaner, clearer image.

To summarize, the goal of identifying calcium by spectral means was successful. The final image was then further enhanced with non-photorealistic shading to provide greater clarity over the image.

5.5 Benchmarking

The performance of MARSCTExplorer heavily relies on the complexity of the intermixing algorithms applied. However, a basic benchmark test can be performed to compare between computer systems if the algorithm is kept constant. Two computer systems were compared over three configurations.

The first system is the *Development System* with an Intel Core2 Quad, 3GB RAM and an NVIDIA GTX 275 graphics card. The second system is the *Visualization System* with an intel i7 920, 12GB RAM, and an NVIDIA

Table 5.2: MARSCTExplorer performance (fps) for two platforms and active stereo (fps per eye) using two resolutions of the Mouse12 dataset.

Computer Systems	**512x512x432**		**1024x1024x432**	
	Single Bin	All Bins	Single Bin	All Bins
Development	25	25	-	-
Visualization	20	20	11	11
Stereo	6	6	4	4

Quadro 5800 graphics card. The visualization system also supports active *Stereo* via NVIDIA's 3D Vision technology providing the third configuration.

The test was performed on two resolutions of the Mouse12 dataset at $512 \times 512 \times 432$ and $1024 \times 1024 \times 432$. The test first used a single energy bin and was later repeated with all four energy bins. The results are shown in Table 5.2.

In all cases, interactive frame rates were achieved. The visualization system easily supported the larger resolution due to the Quadro's memory size. The single and four energy bin cases showed identical performance validating that the vectorization system in the shaders were successful.

An interesting result is that the development system outperformed the visualization system. This can be explained by the specification of the quadro 5800 vs the GTX 275 where the GTX275 has a higher clock speed.

5.6 Summary of Results

The results show that MARSCTExplorer, spectral CT studies can be successfully visualized. For each spectral CT study, suitable intermixing algorithms were found and implemented in real-time to achieve the studies goals. The success with each spectral CT study and the clarity of the results prove that intermixing is a suitable technique for visualizing spectral CT data. In addition, the algorithms found demonstrate processes which can easily be adopted for future studies.

Chapter VI

Conclusion

The goal of this research was to set the groundwork for future user friendly visualization of spectral CT data. This included the development of an intermixing framework and a visualization application called MARSCTExplorer.

6.1 Intermixing Framework

The design of the intermixing framework was developed over three iterations. The final design uses dynamic shader generation to allow the construction of intermixing algorithms from 15 generic operators. The algorithm construction is achieved through an object-oriented system based on operand trees. The resulting expressions are then applied via five data channels including the sample level channel (geometry) and the four illumination level channels (material).

6.2 MARSCTExplorer

The intermixing framework became part of an application called MARSCTExplorer. In addition, MARSCTExplorer also implemented sufficient tools to compare the intermixing framework with non-spectral visualization as well as to navigate through all results. The result is that MARSCTExplorer is a complete research-based spectral CT visualization application.

MARSCTExplorer also contains some extra features including a CUDA accelerated NLM filter and a CUBLAS accelerated PCA calculation utility. The NLM filter is an advanced denoising algorithm which preserves boundaries such that there is no visible data loss. The PCA utility is a unique data analysis tool which is calculated during run-time and then applied in real-time.

6.3 Results

The evaluation of the completed system involved four spectral CT case studies and a benchmarking test. The case studies represented a variety of medical applications and each contained materials difficult to identify or distinguish with non-spectral CT means.

The results from the case studies produce a number of useful outcomes. Firstly, the results demonstrate the value of spectral CT as a medical imaging modality. Secondly, the results demonstrate that the intermixing framework can successfully visualize spectral CT data. Thirdly, the algorithms constructed are based on simple principles and can be applied to future spectral CT studies. Lastly, the results contain a variety of rendering styles all produced with the intermixing framework demonstrating the flexibility of the system.

The benchmarking test compared the frame rates for a basic intermixing algorithm between two machines over three conditions. The results confirm that MARSCTExplorer functions at interactive frame rates.

The results produced with MARSCTExplorer show that this research was successful in its endeavours. The system described in this reseach is a

suitable tool for visualizing spectral CT data and has potential to become an important component in future user friendly visualization software.

6.4 Future Work

While the results achieved with MARSCTExplorer were successful, there is an important limitation of the system designed. The vectorization of GLSL limits the intermixing framework to vectors of size 4 without performance loss. Replacing the volume rendering engine with a CUDA or openCL based implementation would allow the intermixing framework to function with vectors of size n without severe performance loss.

Material extraction was the priority for this research. Future research could attempt to optimize the volume geometries based on the spectral information. In addition, there is a good chance that segmentation algorithms can be vastly improved between energy bins. This would potentially allow faster and more accurate isolation of objects in spectral CT studies.

As spectral CT hardware, knowledge and procedures improve with time, the potential uses of the resulting data will increase and the resulting visualization capabilities will advance. The research of this thesis sets the foundation for visualizing spectral CT data and it is expected that the results and contributions of the research will significantly aid future work.

References

[1] J. Woodring and H.-W. Shen, "Multi-variate, time varying, and comparative visualization with contextual cues," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 909–916, 2006.

[2] N. J. Pelc, "Dual energy ct: Physics principals."

[3] "Medipix." Website.

[4] W. Cai and G. Sakas, "Data intermixing and multi-volume rendering," *Computer Graphics Forum*, vol. 18, no. 3, pp. 359–368, 1999.

[5] G. Hounsfield, "Method and apparatus for measuring x- or Îş-radiation absorption or transmission at plural angles and analyzing the data," December 1973.

[6] J. T. Bushberg, J. A. Seibert, E. M. Leidholt, and J. M. Boone, *The Essential Physics for Medical Imaging"*. Lippincott Williams and Wilkins, 2nd ed., 2002.

[7] R. Zainon, A. P. H. Butler, N. J. Cook, J. S. Butzer, N. Schleich, N. D. Ruiter, L. Tlustos, M. J. Clark, R.Heinz, and P. H. Butler, "Construction and operation of the mars-ct scanner," *Internetworking Indonesia Journal*, vol. 2, pp. 2–10, 2010.

[8] R. Ballabriga, M. Campbell, E. Heijne, X. Llopart, and L. Tlustos, "The medipix3 prototype, a pixel readout chip working in single photon counting mode with improved spectrometric performance," vol. 6, pp. 3557 –3561, 29 2006-nov. 1 2006.

[9] "Octopus version 8.2, server-client tomography reconstruction software."

[10] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel, *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006.

[11] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski, "Advanced illumination techniques for gpu-based volume raycasting," in *SIGGRAPH '09: ACM SIGGRAPH 2009 Courses*, (New York, NY, USA), pp. 1–166, ACM, 2009.

[12] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 163–169, August 1987.

[13] S. Schaefer and J. Warren, "Dual marching cubes: Primal contouring of dual grids," in *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, (Washington, DC, USA), pp. 70–76, IEEE Computer Society, 2004.

[14] N. Pan, H. Liu, N. de Ruiter, and R. Grasset, "Improving the image quality of spectral ct volume rendering," in *IVCNZ09*, pp. 203–208, 2009.

[15] R. Bürger and H. Hauser, "Visualization of multi-variate scientific data,"

in *EuroGraphics 2007 State of the Art Reports (STARs)*, pp. 117–134, 2007.

[16] M. J. Ackerman, "Visible human project," in *The Visible Human Project Conference*, United States National Library of Medicine, October 1996.

[17] A. Ghosh, P. Prabhu, A. E. Kaufman, and K. Mueller, "Hardware assisted multichannel volume rendering," in *Hardware Assisted Multichannel Volume Rendering*, 2003.

[18] C. A. Pelizzari, G. T. Chen, D. R. Spelbring, R. R. Weichselbaum, and C. C. T., "Accurate three-dimensional registration of ct, pet, and/or mr images of the brain.," *J. Comp. Assis. Tomogr*, vol. 13, pp. 20–26, 1989.

[19] M. Ferre, A. Puig, and D. Tost, "A framework for fusion methods and rendering techniques of multimodal volume data: Research articles," *Comput. Animat. Virtual Worlds*, vol. 15, no. 2, pp. 63–77, 2004.

[20] X. Yuan, M. X. Nguyen, B. Chen, and D. H. Porter, "Hdr volvis: High dynamic range volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 433–445, 2006.

[21] H. J. Noordmans, H. T. van der Voort, and A. W. Smeulders, "Spectral volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, pp. 196–207, 2000.

[22] A. Buades and B. Coll, "A non-local algorithm for image denoising," *In CVPR*, vol. 2, pp. 60–65, 2005.

[23] M. Chen and J. V. Tucker, "Constructive volume geometry," *Computer Graphics Forum*, vol. 19, pp. 281–293, 2000.

[24] M. Segal and K. Akeley, *The OpenGL Graphics System: A SpeciiňĄcation (Version 3.2 (Core ProoïňĄle))*. Khronos Group, 3.2 ed., December 2009.

[25] J. Kessenich, *The OpenGL Âő Shading Language*. Khronos Group, 1.50 ed., April 2009. Revision 1.1.

[26] F. Röβler, R. P. Botchen, and T. Ertl, "Dynamic shader generation for gpu-based multi-volume ray casting," *IEEE Comput. Graph. Appl.*, vol. 28, no. 5, pp. 66–77, 2008.

[27] M. Andrecut, "Parallel gpu implementation of iterative pca algorithms," *ArXiv e-prints*, Nov. 2008.

Appendix A

MARSCTExplorer Features

MARSCTExplorer is a research based volumetric visualization application. The features supported by MARSCTExplorer are as follows.

- Volume file types supported
  - Non-compressed DICOM (.dcm).
  - Tagged Image Format (.tif).
- Data types supported
  - Any scalar volumetric data types.
  - Any scalar multi-variate data types.
  - Spectral CT data.
- Rendering methods supported
  - Direct volume rendering (Raycasting).
  - Iso-surfacing.
  - Maximum Intensity Projection.
  - OpenGL 2D slice viewer.
- Tools provided

  - Up to eight views over two screens.
  - Pan/Scale/Rotate.
  - Iterations per ray control.
  - Upper and Lower Thresholding.
  - Blinn-Phong Lighting (sourced from camera).
  - Brightness/Contrast/Gamma.
  - Transfer Functions (post and pre-integrated classification).
  - Axis-based volume clipping.
  - Synchronization of views and geometries.
  - RGBA and HSVA illumination models.
  - Intermixing Framework for registered datasets including
    * Real-time sample and illumination level intermixing.
    * 15 Operators based on elementarty math for scalar and vector data.
    * Stuctured algorithm construction system based on operators and operands.
    * Operand support for volumetric datasets, constants, operators, and transfer functions.
    * Real-time previewing of algorithms.
  - CUBLAS accelerated PCA calculation tool for real-time PCA.
  - CUDA accelerated NLM denoising filter.

- Miscellaneous features

– Basic registration using scaling and translation.

– Saving for all properties applied to a MARSCTExplorer study.

Appendix B

Additional Results

This chapter provides some additional results produced with MARSCTExplorer. These studies allow the quality of MARSCTExplorer to be compared to other visualization systems, reveal how MARSCTExplorer fares when visualizing other data types, and also provide additional insights into the capabilities of MARSCTExplorer as a visualization application.

Figure B.1: Visualization of the first bonsai tree using boolean masks to separate the leaves, trunk and dirt.

Figure B.2: Visualization of the second bonsai tree using boolean masks to separate the leaves, trunk and dirt.

B.1 Visualization of Standard Datasets

The volume library accessible from www.volvis.org has become a standard for volume rendering software. Most literary works on volume visualization will include at least one rendering of a volume from this website. This section demonstrates the restuls achieved for a two of the three bonsai tree datasets.

The bonsai tree datasets are a well known CT scans from the University of Erlangen. There are three 12-bit datasets of small bonsai trees. Figs. B.1 and B.2 show the results achieved with MARSCTExplorer for two of the three trees.

The leaves, trunk, and dirt components were separated with boolean

masks using the step function. Constants were then applied to control the density, hue, saturation, and value of each component. The result is complete control over the colour and visibility of the volume.

B.2 PET-CT Study of Prostate Cancer

The results in Fig. B.4 and Fig. B.5 visualize a PET-CT study to demonstrate that MARSCTExplorer can also visualize multi-modal datasets. This study is of a male with prostate cancer which has spread to the lungs. The goal is to provide a simple overview of the subject and in particular, the neoplasms. A common visualization technique for PET-CT is to simply apply PET as a colour map on top of a CT geometry. This is shown in Fig. B.3.

Four steps were used to visualize the study. Firstly, a simple scale and offset registration system aligns the two volumes. Secondly, boolean masks

Figure B.3: A single slice of a typical PET-CT result. The PET provides a colour map of activity on top of a CT volume.

Figure B.4: The result MARSCTExplorer achieved in visualizing the PET-CT study.

define three important regions including the skin boundary and the organ boundaries from the CT volume, and the neoplasm from the PET volume. Thirdly, the colour scheme is based on the PET volume using constants to provide additional control over the colours for the skin and organ sections. Lastly, the outlining is achieved by setting the ambience, diffusion, and specularity to zero.

Fig. B.5 also clips the volume to better expose the critical neoplasm in the left lung. The result is a clear overview of the location and geometries of the neoplasms in the subject.

Figure B.5: Continuation of Fig. B.4 using clipping to better reveal the important geometry.

Appendix C

Sample Files for MARSCTExplorer

This chapter provides examples of the files associated with MARSCTExplorer. This includes the study file which stores all settings and a sample raycasting fragment shader which is dynamically generated.

C.1 MARSCTExplorer Studies

The MARSCTExplorer study file contains all relevant data and settings. This includes pointers to transfer function files and energy bin directories as well as explicit objects for MARSOperators, MARSConstants, MARSControls, MARSGeometries, MARSStates, and MARSPipelines. Fig. C.1 shows an example study file.

The file uses a simple scripting language which encapsulates each object within a description such as "'%ENERGYBINS"'. This is a flexible system which allows each object to be found quickly and also assists with loading multiple objects with different settings.

For example, if a few of the volumes to be included in the study have identical settings such as spectral CT datasets, then only one energy bin section is required. All volumes will be packed into textures as appropriate. Alternatively, if some settings do changes between volumes such as with multi-modal data which could have different data types, sizes, or registration settings, then multiple energy bin sections can be defined for each volume.

```
%CONTROLLERS
Axial,0.5,3,0.5,1,0,0,0,0,0,0,0,0,1,
%
%GEOMETRIES
/// Ratio.
1,
/// Geometries.
Geometry,0,1,0,1,0,1,
%
%STATES
State,1,0,1,1,0,1,1,0.5,1,1,
%
%CONSTANTS
Constant,0,1,1,
%
%ENERGYBINS
/// Slices.
1,256,256,110,
/// Registration.
1,1,1,0,0,0,
/// Dynamic Range.
0,255,
/// File Type.
.tif,
/// Energy bins.
Engine,
%
%FUNCTIONS
Engine_Dense_Plate,V;0;0,
%
%OPERATORS
%
%PIPELINES
Pipeline,0,V;0;0,F;0;0,F;0;1,F;0;2,F;0;3,
%
%FINISH

```

Figure C.1: A study file records all objects and settings from a MARSCT-Explorer study using a basic, yet flexible scripting language.

Each volume will then be correctly loaded into their own texture.

C.2 Raycasting Fragment Shader

The fragment shader of the rendering algorithms is dynamically generated. The basic form is illustrated in Fig. C.2 which is based on the raycasting algorithim. There are four main areas where code is generated including the uniform list, variable declarations, sample level intermixing, and illumination level intermixing. All other parts of the fragment shader are stored as static code snippets.

The other shader files are stored externally to the software and loaded during run-time. These are common to all rendering algorithms. The first set of shaders includes a vertex and fragment shader to capture the bounding geoemtry's front face. This is used as the starting coordinate. If the camera is inside the volume then the camera coordinate is used instead.

The final shader is the vertex shader for the main rendering pass. This is paired with the dynamically generated fragment shader. The vertex shader captures the back face of the bounding geometry as the ending coordinate of the ray. The results from the first pass are retrieved via a texture mapped to the screen.

```
/** Volume Fragment Shader
**/

/// Fragment I/O
Dynamic Uniform                                  (Generated Code)
Static Uniforms and Attributes                     (Code Snippet)

/// Fragment Start
int main() {
  // RaySetup
  Setup Ray using Hardware Rasterization           (Code Snippet)
  Intermixing Variable Declaration               (Generated Code)
  // Ray Traversal
  for( C = Samples; C > 0.0 && Alpha < 0.98; --C ) {
    // Sample Level
    Intermixing Variable Definition              (Generated Code)
    Intermixing Variable Application             (Generated Code)
    // Empty Space Skipping
    Check Threshold Boundaries {                   (Code Snippet)
      // Illumination Level
      Intermixing Variable Definition            (Generated Code)
      Intermixing Variable Application           (Generated Code)
      // Accumulation Level (Compositing)
      Accumulate Ray Values                        (Code Snippet)
    }
  }
  // Image Level
  Brightness, Contrast and Gamma                   (Code Snippet)
  // Fragment Out
  Fragment = Colour;                               (Code Snippet)
}
```

Figure C.2: The dynamically generated shader is formed from static code snippets and dynamically generated code snippets.